REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Sep 95	SBIR Phase I Final Report

4. TITLE AND SUBTITLE
Interactive Augmentation of Computer Generated Force Behavior Based on Cooperative and Reinforcement Learning

5. FUNDING NUMBERS
M67004-94-C-0055

6. AUTHOR(S)
David A. Handelman
Stephen H. Lane

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
KATrix Inc.
330 Alexander Street
Princeton, NJ 08540

8. PERFORMING ORGANIZATION REPORT NUMBER
TR95-0901

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army STRICOM
Attn:AMSTI-EE (Admiral Piper)
12350 Research Parkway
Orlando, FL 32826

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DTIC
ELECTE
NOV 01 1995
B

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
PUBLIC UNLIMITED

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

State-of-the-art Computer Generated Forces (CGFs) are predictable, non-adaptive, behaviorally distinguishable from manned simulators, and once "figured out," can be outsmarted by manned simulator crews. KATrix's NeurRule™ Technology integrates neural networks with rule-based systems, enabling SAF systems to continuously learn from manned simulators, human instructors, and its own mistakes. This Technology will make CGF behaviors more realistic, and will save time and money by permitting Battle Trainers to modify CGF behavior without the intervention of programmers. Phase I of this project has produced a design methodology for augmenting computer generated force behavior with the NeurRule™ Technology concepts of cooperative and reinforcement learning. The Phase I results indicate that 1) Intelligent CGFs can improve task performance through on-line learning, utilizing information from both supportive and adversarial sources, and 2) the NeurRule™ Technology can coexist with existing SAF software such as modSAF and SAFDI. Phase II will extend these results by enabling SAF operators to interactively develop, through use of an appropriate graphical user interface, CGF behaviors specific to dismounted-infantry operations in urban environments.

14. SUBJECT TERMS
Computer Generated Forces, Machine Learning
Distributed Interactive Simulation, Neural Network

15. NUMBER OF PAGES
52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Interactive Augmentation of Computer Generated Force Behavior Based on Cooperative and Reinforcement Learning

SBIR A93-320 Phase I Final Report

KATrix Inc.
330 Alexander Street
Princeton, NJ 08540
(609) 921-7544

19951027 110

Interactive Augmentation of Computer Generated Force Behavior Based on Cooperative and Reinforcement Learning

SBIR A93-320 Phase I Final Report

Summary

State-of-the-art Computer Generated Forces (CGFs) are predictable, non-adaptive, behaviorally distinguishable from manned simulators, and once "figured out," can be outsmarted by manned simulator crews. Currently, the only remedy for these performance shortcomings is a costly rewrite of the behavioral software model. We propose a hybrid CGF control architecture that integrates neural networks with rule-based systems to enable human-like learning, allowing a SAF system to continuously learn from manned simulators, human instructors, and its own mistakes. This technology will make CGF behaviors more realistic, and will save time and money by permitting Battle Trainers to modify CGF behavior without the intervention of programmers. Phase I of this project has produced a design methodology for augmenting computer generated force behavior with the NeurRule Technology concepts of cooperative and reinforcement learning. The Phase I results indicate that 1) the resulting Intelligent CGFs can improve task performance through on-line learning, utilizing information from both supportive and adversarial sources, and 2) the NeurRule Technology can coexist with existing SAF software such as modSAF and SAFDI. Phase II will extend these results by enabling SAF operators to develop, through use of an appropriate graphical user interface, CGF behaviors specific to dismounted-infantry operations in an Urban Environment. One of the main goals of the Phase II effort will be to demonstrate that SAF operators can gradually shift their attention from low-level control tasks to high-level coordinated plans of action as Smart Opponent and Virtual Teammate CGFs gain competence by learning relevant urban assault tactics and evasive maneuvers.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special

Table of Contents

1. PROBLEM STATEMENT

1.1 SIGNIFICANCE OF THE PROBLEM

State-of-the-art Semi-Automated Forces (SAF) systems in use by the Army today produce behaviors that are predictable, non-adaptable, and distinguishable from those of manned forces. In addition, the functionality of these SAF systems is based on operations consistent with the theatres of war existing at the time of their design. A SAF system which could continuously learn from trainees, human instructors, or its own mistakes would provide much more challenging and realistic friendly and opposing forces as it could adapt to new threats and tactics as they develop, and keep pace with likely battle scenarios in a changing world. Currently, the only solution to these performance limitations, if they are even possible, is a costly and time consuming rewrite of the Computer Generated Force (CGF) behavioral software model.

1.1.1 Current SAF Systems Don't Do As Well As Manned Forces

Semi-Automated Forces (SAF) systems have been used in Distributed Interactive Simulation to reduce the cost of training by substituting manned Opposing Force (OPFOR) simulators with computer-generated simulations. The Turing Test for Semi-Automated Forces has been that its behavior be indistinguishable from the equivalent manned simulators, from the standpoints of both human observers and computed statistics.

All SAF systems developed to date have been either static rule-based or procedural-based systems which do not have the capability to adapt their behavioral models. Since the depth of complexity of the behavior of a human has never been incorporated into these systems, the resulting behaviors of the CGF entities appear normal in many situations, but break down into bizarre, conspicuous, anomalous behavior in many special cases. This phenomenon has been observed in all SAF systems delivered to-date.

1.1.2 CGF Behavioral Anomalies Jeopardize Simulation Results

When a manned simulator crew detects anomalous behavior in an enemy vehicle, they know that it is a CGF vehicle and not another manned crew. Knowing that the CGF vehicle is not as intelligent, and cannot learn from its mistakes, the manned crew treats the CGF vehicle differently, exploits its obvious shortcomings, and learns how to easily defeat it. Furthermore, the manned crew becomes more adept over time at detecting and defeating CGF vehicles. These unrealistic combat interactions jeopardize the credibility and accuracy of simulation results.

1.1.3 Current Repair Cycle is Costly, Time Consuming, and of Limited Effectiveness

The following procedure has historically occurred:

- A battle observer notices one of these behavioral anomalies, stops the current experiment or training exercise, and informs the SAF Operator of the problem.
- The SAF Operator spends time reproducing and observing the phenomenon, then reports it.
- A procurement activity is initiated to allow software developers to work on the problem.
- The developers iterate and test potential fixes. The more complicated the behavioral model, the longer it takes to find the cause of the anomaly and repair it.
- When that one particular problem is fixed, the software developers create a new software release and the sites undergo the pain and suffering of a software upgrade.

The current repair cycle for these CGF deficiencies is shown on the left side of Fig. 1.1. There is a tremendous amount of lost time, foregone site utilization, and additional software development cost in this cycle. The end result of this repair cycle is usually a custom patch for one or two of these specific anomalies. As these systems get exponentially more complicated, the number of possible anomalies also increases exponentially.

1.1.4 Current SAF Systems Cannot Adapt to New Threats

One of the principal uses of SIMNET and DIS is for combat weapons development. Hypothetical weapon systems are placed on the synthetic battlefield and tested against benchmarked systems to ascertain their contribution to overall force effectiveness. In the real world, an enemy would more than likely change his tactics upon the detection of a new weapon system with unanticipated performance characteristics.

> ***Example:*** A new sensor capable of detecting ground vehicles moving faster than 45 kph is incorporated into friendly force vehicles. The enemy, seeing that the kill rate for vehicles moving faster that 45kph has increased dramatically, changes his tactics and does not exceed 45kph.

Current SAF systems cannot learn from battlefield results, and hence, would never change their behavior and travel less than 45kph. If a new sensor, like the one described, were tested against current SAF systems, the evaluation of its effectiveness couldn't possibly be accurate.

1.2 APPLICATION OF KATRIX NEURRULE™ INTELLIGENT AGENT TECHNOLOGY

In order to properly prepare soldiers for events in a changing world, current SAF systems must be made more realistic, flexible and adaptive. Battle Trainers should be able to create and modify specific CGF behaviors in a timely manner, without the delay and cost normally associated with a procurement cycle involving programmers. KATrix's NeurRule Intelligent Agent Technology, when incorporated into existing SAF software, can provide these desired system capabilities by creating Computer Generated Forces that:

- ❑ Continuously learn behaviors through mimicking examples of others.
- ❑ Modify behaviors in real-time through the adaptation of running software, as opposed to fixing source code off-line.
- ❑ Learn in a "show" and "tell" manner from Battle Trainer as opposed to being coded by a programmer.
- ❑ Continuously learn DURING a battle, just like soldiers.
- ❑ As the software becomes smarter, computational loading actually drops!

The right side of Fig. 1.1 shows what a repair cycle using our NeurRule Technology might look like. The procedure should be as simple as:

- ❑ Battle Trainer notices anomaly.
- ❑ Battle Trainer places CGF in special learning mode.
- ❑ Battle Trainer tunes behavior using a high-level GUI development tool or jumps into a simulator and directly demonstrates correct behavior to SAF system (multiple times if necessary).
- ❑ SAF system learns correct behavior in real-time.

This streamlined repair cycle cuts several organizations out of the loop, saves a tremendous amount of time and money, and performs the correction to the degree and satisfaction of the end user, the Battle Trainer. In addition, the SAF system will continually hone its own behavior, fixing problems BEFORE the Battle Trainer detects them.

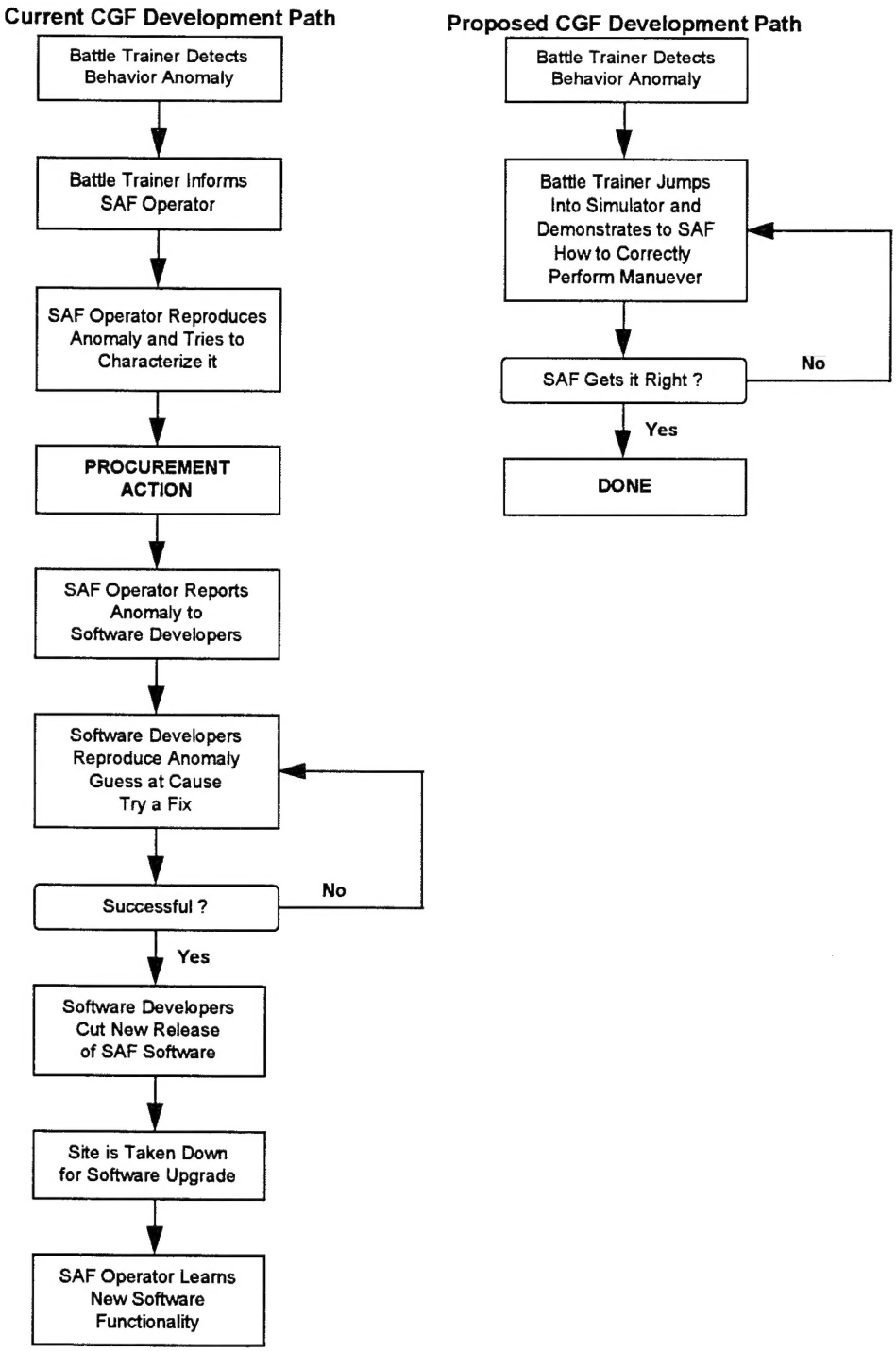


Figure 1.1. *Comparison of current and proposed CGF development paths.*

2. TECHNICAL OBJECTIVES AND APPROACH

2.1 GENERAL PROJECT OBJECTIVES

The general objective of this project is to investigate how Army SAF systems can be made more realistic, flexible and adaptive by enabling Battle Trainers (SAF System Operators) to interactively create and modify complex CGF behaviors on-line in a timely manner, without the delay and cost normally associated with a procurement cycle involving programmers.

The present project includes the following general objectives that will demonstrate the utility of applying the KATrix's NeurRule™ Intelligent Agent Technology to computer generated forces (CGF) in distributed interactive simulations, :

- ❑ Show that CGF models can be made to continuously adapt in real-time.
  - From its own behavior.
  - From its opponents (both successful strategies and mistakes).
- ❑ Show that the model can be taught by a human instructor.
  - Cognitive plans of action.
  - Hand/eye coordination.
- ❑ Show that training is the same as for a human student, and can be done by military personnel instead of a programmer.
- ❑ Show that the model's learning rate and level of competency is user selectable.
- ❑ Show that as the model learns, computational burden actually drops.

2.2 TECHNICAL APPROACH

Our hybrid learning approach to computer generated forces development attempts to copy two key aspects of human skill acquisition: how successful control strategies are conveyed from one human to another, and how processing within the learner changes as performance improves.

Computer generated forces that can continually challenge players with a range of skills, from novice to expert, are extremely rare as the required logic is too time consuming and costly to create. KATrix NeurRule Technology meets this need by enabling a new generation of "smart games" to be developed with computer generated forces that learn and behave based on their experiences with the human player.

2.2.1 General Description of KATrix NeurRule Technology

KATrix NeurRule™ Technology is an outgrowth of research in robotic control theory and artificial intelligence that attempts to copy key features of human motor control and skill acquisition. The core components of the NeurRule™ Technology consist of *Limb Coordination*, *Neural Network*, *Rule-Based Control*, and *Base Agent Libraries* that allow reflexive, goal-directed, and learning behaviors to be added to fully-articulated characters in interactive computer games and virtual reality simulations.

Limb Coordination Library. A unique approach to inverse kinematics especially well-suited toward interactive simulations and computer gaming. Provides real-time, task-oriented control of limb movements and balance in fully interactive human and animal-like articulated figures. Pull forward on a character's hand and it leans back to keep its balance. Throw a ball at its head and it ducks out of the way. By combining multiple low-level Limb Coordination primitives called *Synergies*, designers can achieve sophisticated inverse kinematic behaviors such as walking, running and jumping while minimizing design effort and on-line computational costs.

Neural Network Library. Computational techniques enabling *Smart Opponent*™ and *Virtual Teammate*™ game characters to be created that exhibit human-like learning characteristics. Smart Opponents learn your moves, then use them against you. They get better the more they are played, and learn to defeat players who are not innovative. Virtual Teammates are on your side, and enable players to transfer (download) their hand-eye coordination to an Intelligent Agent in much the same way a coach trains an athlete.

Rule-Based Control Library. Goal-directed inference engine and knowledge base allowing designers to create intelligent behaviors using hierarchical rule-based descriptions of competitive strategies and tactics. Issue the command "attack incoming bogie at 12 o'clock" and Intelligent Agent-based characters decompose and execute goal-directed plans of action automatically as the inference engine searches the task knowledge base.

When rule-based task execution is used in conjunction with neural networks, Intelligent Agents can acquire skills over time through on-line learning.

Agent Class Library. An object-oriented framework for building Intelligent Agents. Includes sensorimotor system utilities enabling Intelligent Agents to focus on objects of interest in the world (such as walls, doors, trees, body parts of other agents, etc.), sense relevant state information (such as range and bearing to the player, location of the nearest door, proximity to food, etc.), generate smooth dynamical responses to control inputs, and execute basic motor tasks (such as Goto, LookAt, AlignWith, Follow, etc.).

2.2.2 NeurRule Intelligent Agent Control Architecture

Built upon the NeurRule Technology Libraries listed above, the NeurRule Intelligent Agent Control Architecture [Handelman & Lane, 1993a,b,c; Handelman, Lane & Gelfand, 1990, 1992, 1993; Lane, Handelman & Gelfand, 1992] depicted in Fig. 2.1 contains both rule-based components and neural networks. The rule-based components of this hybrid controller provide a system designer with a convenient high-level symbolic way of specifying explicit plans, rules of engagement, and control strategies.

Rule-based components continuously invoke a two-stage inferential process:

- The first stage specifies desired behavior through *goal-directed task descriptions*.
- The second stage provides *error-driven feedback control commands* using fuzzy rules, and conventional control algorithms that tend to move the system in the direction of desired behavior.

Neural network components [Albus, 1975; Lane, Handelman & Gelfand, 1992] gradually learn to minimize error-driven commands using a training paradigm called *feedback-error-learning* [Miyamoto et al, 1988]. This learning procedure enables the system to:

- Autonomously learn how to improve task performance, and
- Decrease the amount of computation required by "turning off" error-driven rules when learning has sufficiently improved performance.

KATrix's proprietary neural network architecture differs from most others in two key ways. First, it performs ***local, not global, function approximation***. That means that when it learns something in one context, it is less likely to corrupt things it has already learned in other contexts. It also results in less learning-related computation. Second, our neural networks ***learn on-line.*** This means that unlike most architectures, they don't need a large amount of random batch training samples. Learning occurs as the dynamic system evolves naturally, that is, as the game is played.

Human operator-based error-driven commands also can be accommodated in the NeurRule Control Architecture. Thus, in addition to enabling autonomous learning, the control architecture also permits an operator to train CGFs on-line by

- ***showing it*** using joysticks, datagloves, etc., and
- ***telling it*** using rule-based descriptions

how to improve task performance.

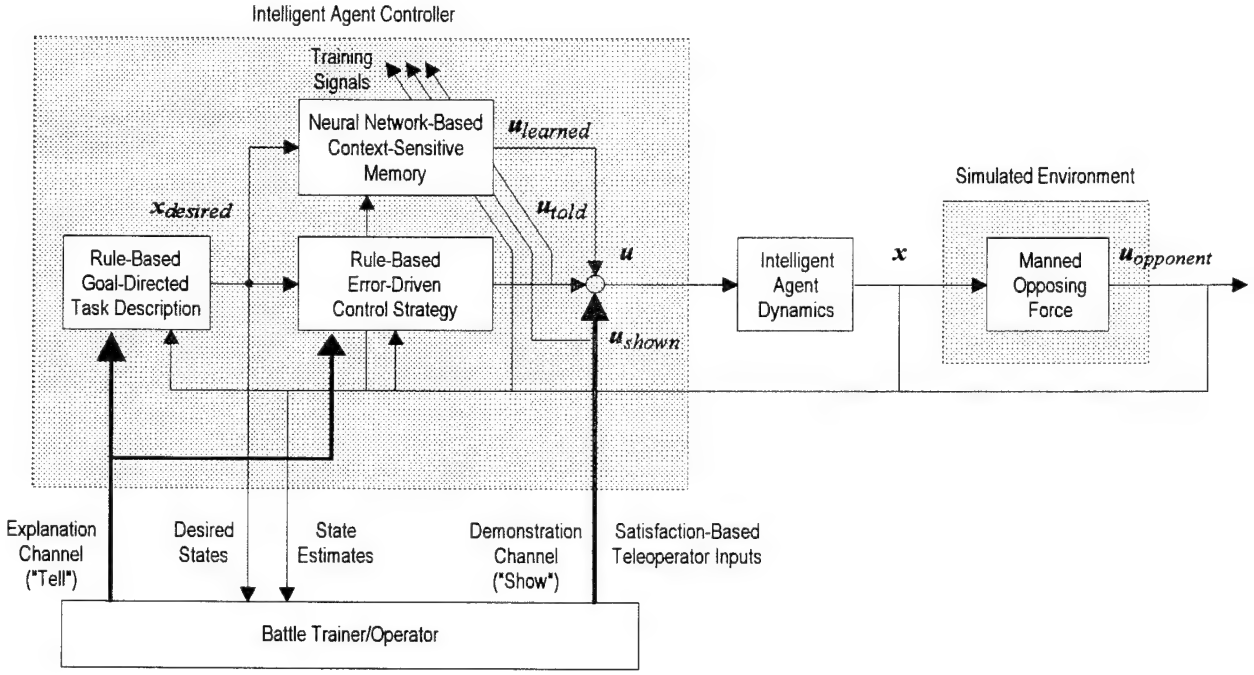


Figure 2.1. *NeurRule™ Intelligent Agent Control Architecture* utilizes three sources of training information. Goal-directed task descriptions specify plans of action and rules of engagement. Rule-based error-driven commands and teleoperator inputs are summed with neural network outputs to control the intelligent agent, and are faded out as the neural networks learn. The hybrid system also learns from opponents by reinforcing in its own task space actions of the opponent that prove successful against it.

2.2.3 Human-to-Machine Skill Transfer through Cooperative Learning

When a CGF is configured as a Virtual Teammate, it is "on your side," and can be trained using cooperative learning to autonomously perform a task using hands-on examples (showing) and rule-based task descriptions (telling) of proper task execution.

As a Virtual Teammate, the CGF and operator work as a team to accomplish a task, and the operator's contribution smoothly diminishes as the system learns. Consequently, we call this form of human-to-machine skill transfer ***cooperative learning***. Virtual Teammate ***skill acquisition*** is associated with the shift from a predominantly feedback-oriented, rule-based representation of control to a predominantly feedforward, neural network-based form. Analogies can be drawn to the way humans acquire skill, highlighting features we believe are essential to obtaining truly autonomous CGFs.

- A limited amount of strategic knowledge provided by rule-based components initiates motion.
- Performance improves incrementally through learning, with inferential problem solving giving way to reflexive motor programs provided by neural networks.

Computational efficiency is intended for areas of the dynamic state-space visited often, as in repetitive maneuvers. Inferential problem solving remains ready, however, to handle infrequently executed tasks, or changes in the intelligent agent or its environment that render previously acquired expertise ineffective. Hence CGFs developed with our NeurRule Technology are highly adaptive and robust.

2.2.4 Human-to-Machine Skill Transfer through Reinforcement Learning

When a CGF is configured as a Smart Opponent, it can be trained using reinforcement (punishment/reward) learning to represent player strategies and tactics that have proven successful against itself.

In this scenario, a CGF configured as a Smart Opponent starts out with general rule-based knowledge about competitive strategies and tactics. As the associated NeurRule controller encounters human behavior, its neural networks learn to copy useful behaviors, and to reinforce actions that prove successful against itself. For example, hiding and attack behaviors exhibited by human opponents can be mimicked and used against them, as well as knowledge concerning how to aim and when to fire.

- These synthetic enemies learn to ***defeat human opponents that continue doing the same thing over and over***. Only by evoking unique behavior (doing something novel) can a human opponent gain a temporary advantage.
- The behaviors learned by the Smart Opponent are unique to the human player to which it is exposed, creating a ***different experience for each participant***.

The learning capability enabled by the NeurRule Technology can be used to inexpensively simulate large numbers of troops in distributed interactive simulations and virtual reality applications. Additionally, because human participants must adapt their behavior as the CGF learns, it becomes an effective training tool.

2.2.5 Summary of Computer Generated Force Learning Behaviors

Computer generated forces that learn and behave based on their experiences with the human player can be developed in a number of ways using various types of Virtual Teammate and Smart Opponent learning.

Virtual Teammate Learning

Player-Based Mimicking. CGF learns to copy a player's behavior as the game is played.

Player-Based Cooperative Learning. Player "steps into" the CGF and incrementally modifies its behavior, downloading his or her hand/eye coordination into the control system of the CGF. Player contributions are steadily phased out as a result of neural network learning.

Smart Opponent Learning

Player-Based Mimicking. CGF learns to copy a player's behavior as the game is played.

Player-Based Reinforcement Learning. CGF learns to reproduce actions used by player that prove successful against itself, such as aiming and firing tasks.

Developer-Based Mimicking. CGF learns to copy a developer's behavior as the developer plays the game during the development process. Learned neural network behaviors are then used to augment existing capabilities.

Developer-Based Cooperative Learning. Developer "steps into" the CGF and incrementally modifies its behavior, downloading his or her hand/eye coordination into the control system of the CGF during the game's development. Learned neural network behaviors are then used to augment existing capabilities.

2.2.6 Sample Learning Scenarios and Implementation Issues

Sample Scenarios

Summarized below are some sample simulation scenarios that can benefit from Smart Opponent and Virtual Teammate learning. The neural networks shown have as inputs simulation state variables that define the context of an existing situation, and as outputs, simulation variables that directly control CGF locomotion and other behaviors. Note any convenient coordinate system may be used to define neural network inputs. For example, although spherical coordinates are shown below, Cartesian coordinates could be used just as easily.

Artillery

You are in a tank battle, facing a platoon of Smart Opponents. You aim your gun turret sideways and up and down, and you choose when to fire. As you fight, the enemy learns to shoot the way you do.

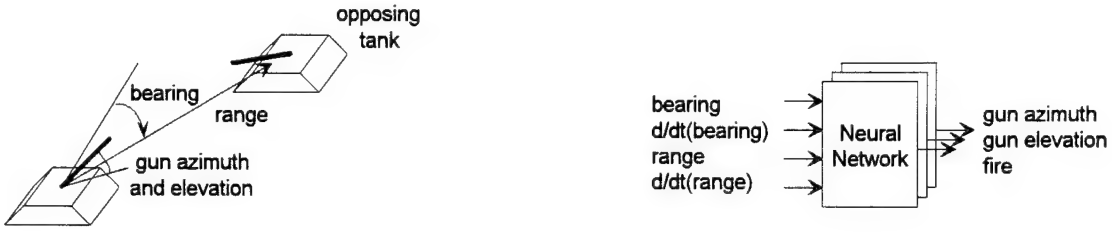


Figure 2.2 *Neural Net configuration for Artillery Scenario*

Gunfight

You are in a gunfight with Smart Opponents that run around and hide behind obstacles. They learn from you how to aim and when to fire, and they don't waste ammunition firing while you're hiding.

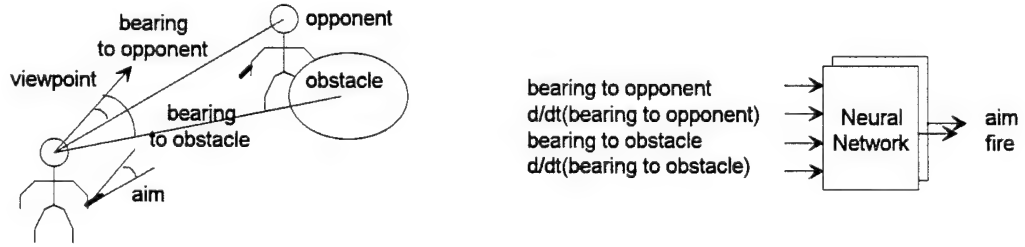


Figure 2.3 *Neural Net configuration for Gunfight Scenario*

FireTeam

You must teach each member of a Virtual FireTeam how to respond when attacked. For example, you show your rifleman how to take up a defensive position and respond with a counterattack. After training, you direct the Team as the Team Leader.

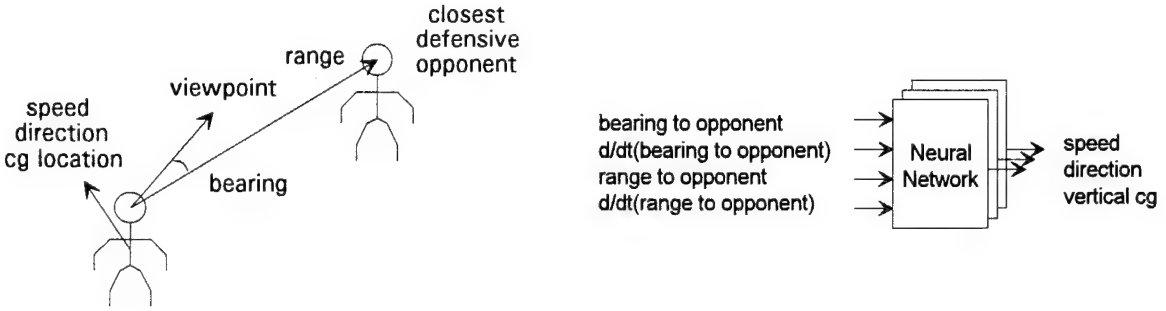


Figure 2.4. *Neural Net configuration for FireTeam Scenario*

Dogfight

You are a pilot engaged in a dogfight. Initially, you can easily shake the Smart Opponent on your tail, but as the game evolves, he gets better and better at tracking you ("Smart Tracking"). You can also teach a Virtual Teammate wingman how to fly next to you in formation.

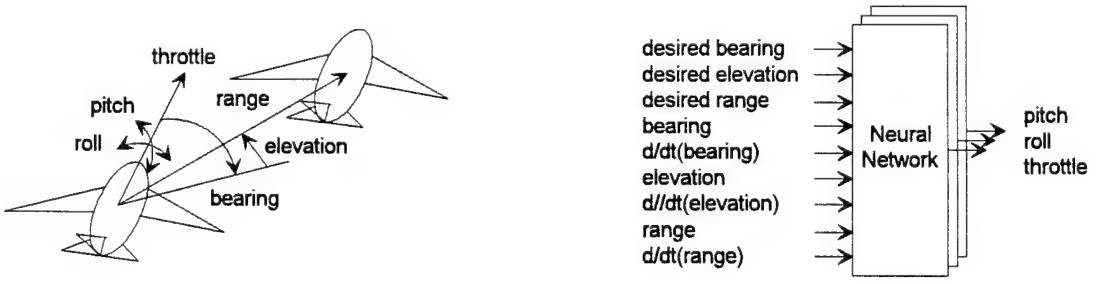


Figure 2.5. *Neural Net configuration for Dogfight Scenario*

We have implemented real-time simulations of systems that demonstrate all of the capabilities mentioned above [see Section 3.4, Related Work].

Typical Implementation Costs

Tables 2.1 and 2.2 list typical implementation costs associated with the sample scenarios outlined above. Note that learned behaviors can be shared between common CGFs within a game. Furthermore, a single neural network can be used to encode the behavior of multiple CGFs by adding extra network inputs that distinguish the CGFs.

Table 2.3. *Typical implementation costs associated with sample scenarios.* Memory, code, and execution times are for C code on an 25-MHz 486 PC with math co-processor.

Scenario	Neural Network Inputs	Neural Network Outputs	Shared Code Size (bytes)	Total Data Size (bytes)	Total Recall Time (msec)	Total Recall&Learn Time (msec)
Artillery	4	3	20 k	48 k	1.2	1.9
Gunfight	4	2	20 k	32 k	1.1	1.5
FireTeam	4	3	20 k	48 k	1.2	1.9
Dogfight	9	3	20 k	48 k	1.9	2.5

Table 2.4. *Typical implementation costs associated with off-line and on-line learning.* Memory, code, and execution times are for C code on an 25-MHz 486 PC with math co-processor.

Time of Learning	Shared Code Size (bytes)	Per Output Data Size (bytes)	Locate 4-Input Space Time (msec)	Per Output Recall Time (msec)	Per Output Recall&Learn Time (msec)
Off-Line	20 k	16 k	0.69	0.16	0.38
On-Line	20 k	16 k	0.69	0.16	0.38

3. PHASE I RESULTS

Phase I of this project has produced a design methodology for augmenting computer generated force behavior with the NeurRule Technology concepts of cooperative and reinforcement learning. The Phase I results indicate that the resulting Intelligent CGFs can improve task performance through on-line learning, utilizing information from both supportive and adversarial sources, and that the NeurRule Technology can coexist with existing SAF software such as modSAF and SAFDI.

3.1 PHASE I TECHNICAL OBJECTIVES

The Phase I work investigated the development of intelligent computer generated force behaviors using KATrix's NeurRule Technology. The feasibility of this approach has been determined through accomplishment of the following Phase I technical objectives:

- Demonstrating that a system operator can endow intelligent agents with warfighting strategies, tactics, and sensorimotor skill (hand/eye coordination) on-line by ***"showing" and "telling" them how to behave***, in an incremental manner similar to a coach training an athlete.

- Demonstrating that the resulting smart opponents can ***learn a player's moves and use them against him***, thereby presenting a unique scenario for each player and encouraging player ingenuity.

- Demonstrating that the performance of computer generated forces can be made to ***improve significantly over time*** by autonomously shifting from a deliberative form of processing to a reactive one.

The major results of the Phase I effort are summarized below in the following sections:

3.2 Specification of Intelligent CGF Behaviors

3.3 Performance of Intelligent CGF Behaviors

3.4 Integration of Intelligent CGF Behaviors with Existing SAF Systems

3.2 SPECIFICATION OF INTELLIGENT CGF BEHAVIORS

To be considered intelligent, CGF behavior must exhibit the following properties:

- From the soldier's point-of-view, CGF behavior should be indistinguishable from that of real soldiers.
- From the trainer's point-of-view, the behavioral interactions between soldiers and CGFs are indistinguishable from battlefield performance.
- CGFs can perform any mission or maneuver actual soldiers can.

In other words, in an ideal system, the soldiers think they're going against other humans, the trainers think they're seeing a real battle, not just a game, and any battle can be simulated. To achieve this capability, CGFs must be able to:

- Exercise a wide range of task knowledge
- Adapt to changes in the environment and opposing force performance

During Phase I of this project, a methodology was developed whereby a developer of CGF behavior (a battle trainer or SAF operator) could use integrated rule-based and neural net-based control technology to enable these capabilities.

3.2.1 Intelligent CGF Behaviors - Basic Assumptions

Basic assumptions include the ability of the CGF developer to know exactly what should be done during a maneuver, and his/her ability to measure success. Therefore it was assumed that the Battle Trainer/SAF Operator:

- provides rules of engagement (task rules)
- provides rules for improvement (correction rules)
- provides hands-on examples of improvement (operator inputs)
- can recognize acceptable performance.

Our approach to CGF development requires that behaviors be defined relative to a "focus of attention". The ability to focus is a key aspect of many human motor control tasks. A baseball player keeps his eye on the ball, a driver keeps his eyes on the road. We require that behaviors be focused, and devote resources to both the implementation of a maneuver and to the focusing on objects of interest in the world relevant to that maneuver.

3.2.2 Intelligent CGF Development Cycle

Figure 3.1 outlines the design procedure developed during Phase I of the Project for the construction of learning behaviors in Intelligent Computer Generated Forces.

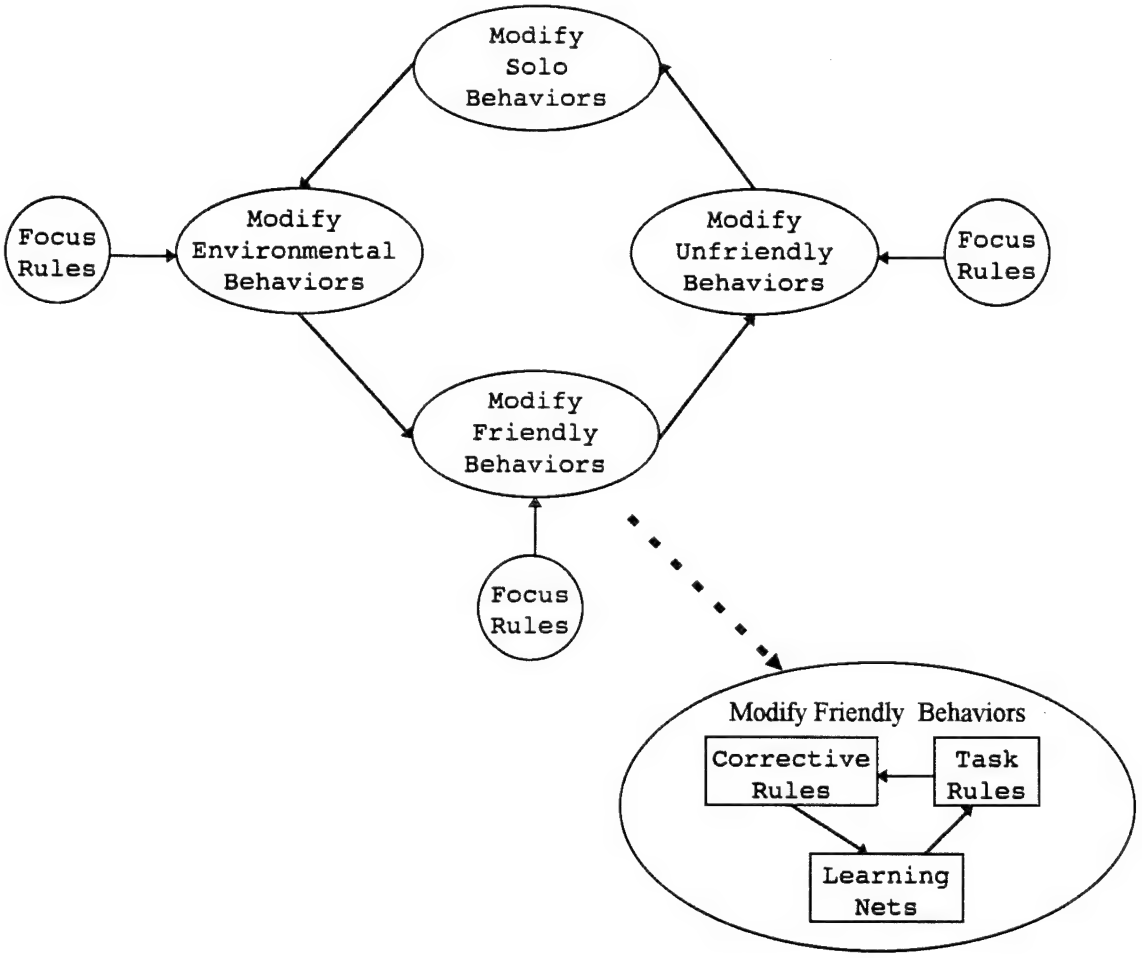


Figure 3.1: *Development of explicit task knowledge.*

First, solo behaviors are created, then those dealing with environmental interactions, and then those involving friendly and unfriendly opposing forces.

Solo Behaviors

Solo behaviors are tasks that relate to one's own body. The ability to move in a specified direction at a specified speed, to turn at a specified rate, and to assume desired postures are all solo behaviors. Our technology library includes such low-level vehicle and body functions.

Environmental Behaviors

Environmental behaviors are maneuvers performed with respect to objects that in general don't move. If objects of interest in the world include trees, bridges, and buildings, then the ability to GoTo a bridge or AlignWith a doorway represents an environmental behavior.

Focus Rules

Most behaviors are referenced to something being focused on. When we design a behavior to cross a bridge, we start out by assuming an object of class bridge is being focused on, then we design the behavior. We can then cross any bridge within that class, but first we must be focusing on it. The Focus Rules decide what to focus on. Given a choice of two bridges to cross, they would choose which one to cross, then the "cross bridge" environmental behavior rules and nets would perform the task.

Friendly Behaviors

Friendly behaviors involve friendly forces. Once a friendly is being focused on, low-level functions such as GoToMate, LookAtMate, and AlignWithMate can be used to move with respect to the object of interest. Higher-level rules can be used to implement formations and more complex tasks. The key is that the behavior is generic with respect to a chosen class of friendly force, and is instantiated with particulars at run-time.

Unfriendly Behaviors

Unfriendly behaviors involve opposing forces. In terms of low-level navigation, they can have much in common with friendly behaviors, such as the ability to GoToEnemy, LookAtEnemy, and AlignWithEnemy. In this case, a simple change of focus (from friendly item to opposing item) can tap into a rich library of fundamental behaviors.

Within each of the above groups, behaviors are built from the building blocks of task rules, corrective rules, and learning neural nets as shown in Fig. 3.1. The development process is cyclic because it may be discovered during the construction of a high-level behavior, such as QuietlySurroundEnemy, that a new lower-level solo behavior, such as MoveQuietly, must be created. Combined with the concept of focusing, the behavior groups help organize the type of knowledge utilized by computer generated forces.

3.3 Performance of Intelligent CGF Behaviors

This section presents results associated with applying the intelligent CGF development methodology of Section 3.2.3 to two battle scenarios. Sections 3.5 and 4 describe how the Phase II effort will further develop the methodology and integrate real battle trainers.

3.3.1 Forest Chase Battle Scenario

The experiments discussed here represent results from a forest chase battle scenario involving a tank chase through a dense forest. Learning technology enables the following types of CGF behavior:

- A single unit CGF chases a retreating tank through the forest.
- The retreating tank tries to shake the CGF through evasive maneuvers.
- The CGF learns to reduce tracking error over time using rules and teleoperator inputs.
- The CGF learns to reproduce the retreating tank's evasive maneuvers.
- Additional members of the chase also reproduce the retreating tank's evasive maneuvers, showing the applicability to group behaviors.

As shown in Fig. 3.2, the forest consists of rectangular poles and the tanks are represented as triangular hovercraft (jets). Although the graphical respresentations used are simple, the results in the following sections demonstrate the utility of the NeurRule Technology in CGF applications.

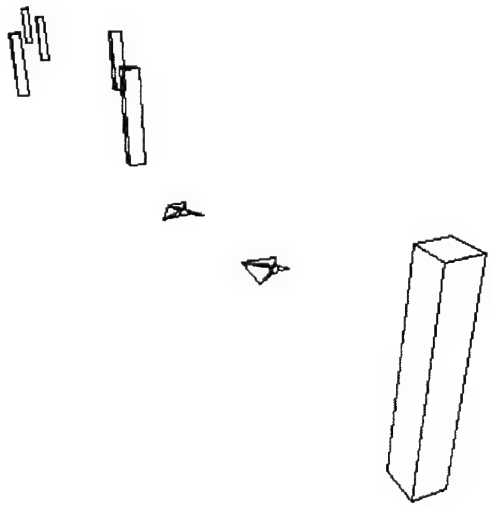


Figure 3.2. *The simplified forest chase environment.*

CGF Performance with No Learning

A single unit CGF chases a retreating tank through the forest. The retreating tank tries to shake the CGF through evasive maneuvers.

Obstacle avoidance, no learning

In Experiment 1a, the goal of the maneuver is to dash from one end of the forest to the other while avoiding trees that may get in the way. The jet executes this "dash" behavior with the following rules, which represent a hierarchical, goal-directed description of the task.

```
To dash,
  If you're too close to a tree, avoid it.
  Otherwise, go from pole to pole.

    To avoid a tree,
      stop and turn until you're not facing it.

    To go from pole to pole,
      go to the pole of interest,
      then focus on the other pole.

        To go to the pole of interest,
          first get near the pole,
          then go around it.

            To get near the pole,
              look at the pole and move toward it
              until you're within a specified distance from it.

            To go around the pole,
              circle around it,
              leaving it to the side it started on,
              until you've reversed course.
```

Each of these rules encodes a modular piece of knowledge. When combined, they perform a complex task. Each rule falls into an explicit task functional block shown in Fig. 3.1. The third rule is an environmental behavior focus rule, whereas the remaining rules are environmental behavior task rules. The NeurRule Rule-Based Control Library enables these rules to include embedded C code.

As shown in Fig. 3.4a, the jet performs the intended task - going from pole to pole - using a rule-based, goal-directed task description. However, it keeps getting stuck near trees. Experiment 1b adds neural network-based learning to improve its performance.

Chasing, no learning

Experiment 2a depicts a second jet trying to chase the first jet. The second jet is executing a "chase" behavior that includes the following rules.

```
To chase,
   If you're too close to a tree, avoid it.
   Otherwise, follow the enemy
   while bypassing trees.
```

```
To follow the enemy,
   go to the specified distance from him,
   look at him, and don't slide sideways.
```

The first rule is part of the CGF "avoid tree" environmental behavior, whereas the second is a task rule within the CGF "follow enemy" unfriendly behavior.

As shown in Fig. 3.7a, the chasing jet performs the task using rules only, but not very well (intentionally). Experiment 2b shows how neural net learning improves the performance.

Formation flying, no learning

Experiment 3a shows two wingmen attempting to fly in formation behind the chasing jet. The "follow" behavior includes the following rules.

```
To follow,
   If you're too close to a tree, avoid it.
   Otherwise, follow your mate
   while bypassing trees.
```

```
To follow your mate,
   go to the specified distance from him,
   align yourself behind and beside him,
   look in the direction he is,
   set your speed equal to his,
   and don't slide sideways.
```

As part of Fig 3.1, the second rule shown above is considered a CGF friendly behavior task rule.

The wingmen perform their task well using only rules until they get close to a tree, at which point they slam on the brakes to avoid it. As shown in Fig. 3.11, the basic tree-bypassing rule doesn't work well enough. Experiment 3b uses neural network learning to fix the problem.

CGF Perfcrmance with Learning

Learning from rule how to avoid obstacles

This experiment improves the "dash" behavior by phasing in the ability to smoothly bypass trees.

```
To dash,
   If you're too close to a tree, avoid it.
   Otherwise, go from pole to pole, and
   bypass trees that look familiar.
```

```
To bypass a tree,
   If you're near the tree and facing it,
      turn and slide away from it.
   If you're near the tree and turning toward it,
      turn and slide away from it.
```

The last rule implements a proportional-derivative control law using fuzzy rules. With respect to Fig 3.1, it is considered a corrective rule within the CGF "bypass tree" environmental behavior. It enables the jet to smoothly bypass trees that look familiar. We use the output of a neural network to provide a measure of familiarity.

Figure 3.3 depicts the neural net used to recognize trees. It represents an environmental behavior learning net within Fig. 3.1. It takes as inputs information representing the present state of the world relevant to recognizing trees, in this case, range and bearing to the nearest tree and their respective rates of change. The neural net outputs a number between 0 and 1 that indicates how familiar the jet is with the closest tree. Its output is initialized to 0 indicating that it knows nothing about trees. During training, the neural net is taught to output a 1 whenever it encounters a tree, so that in the future it will recognize it and invoke the tree-bypassing rule.

The more the jet is exposed to trees, the more familiar they look. When trees look familiar, they are bypassed, reducing the amount of time it takes the jet to go from pole to pole. The behavior snapshots of Fig. 3.4 and the error plots of Fig. 3.5 show how the jet's performance improves over time.

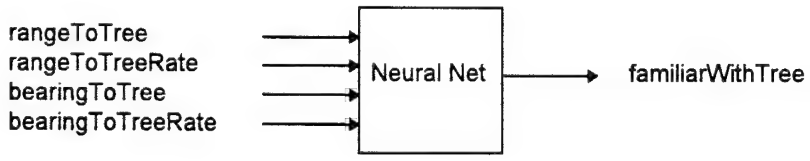


Figure 3.3. *Neural network used to improve obstacle avoidance behavior.*

This type of context-dependent learning, exhibiting learning curves similar to those associated with human skill acquisition, can be used to provide DIS trainees with more-realistic friendly and opposing forces.

(a) *Before learning, 5th attempt.* (b) *After learning, 5th attempt.*

Figure 3.4. *Snapshots of obstacle avoidance behavior before and after learning.*

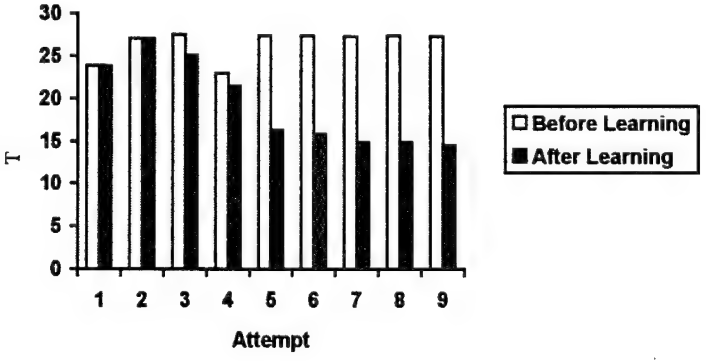


Figure 3.5. *Improvement in obstacle avoidance behavior due to learning.*

Learning from rules how to chase

In Experiments 2a and 2b, the ability of the chasing jet to follow its enemy is proportional to its familiarity with the relative position of the enemy through FollowEnemy rule. In Experiment 2a, with no learning, the chasing jet's familiarity with the enemy is assumed to be a small default value (0.3). In Experiment 2b, the chasing jet's familiarity with its enemy is a learned function provided by a neural network. The inputs and output of the net are depicted in Fig. 3.6. Using the neural net, the chasing jet learns to recognize relative enemy positions. When the enemy is in a state that looks familiar, the jet can follow the enemy more closely using a stronger gain on the low-level tracking behaviors embedded within the FollowEnemy rule.

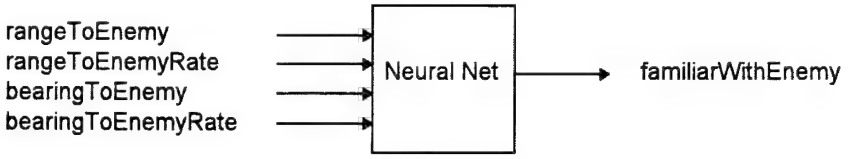
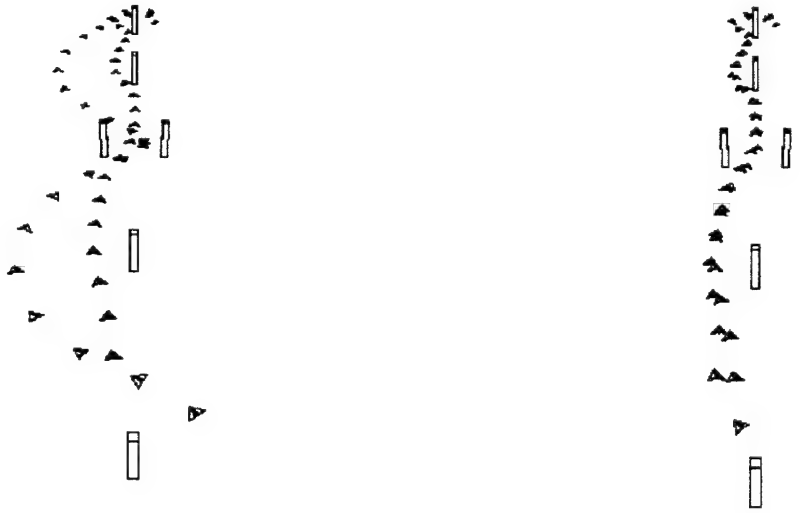


Figure 3.6. *Neural network used to improve chasing behavior.*

The output of the neural net acts as a context-dependent gain that becomes stronger for areas of the state space with which it is familiar. Over time, the jet follows its enemy better and better, as shown in the behavior snapshots of Fig. 3.7 and the range and bearing error plots of Fig. 3.8.



(a) *Before learning, 13th attempt.* (b) *After learning, 13th attempt.*

Figure 3.7. *Snapshots of chasing behavior before and after learning.*

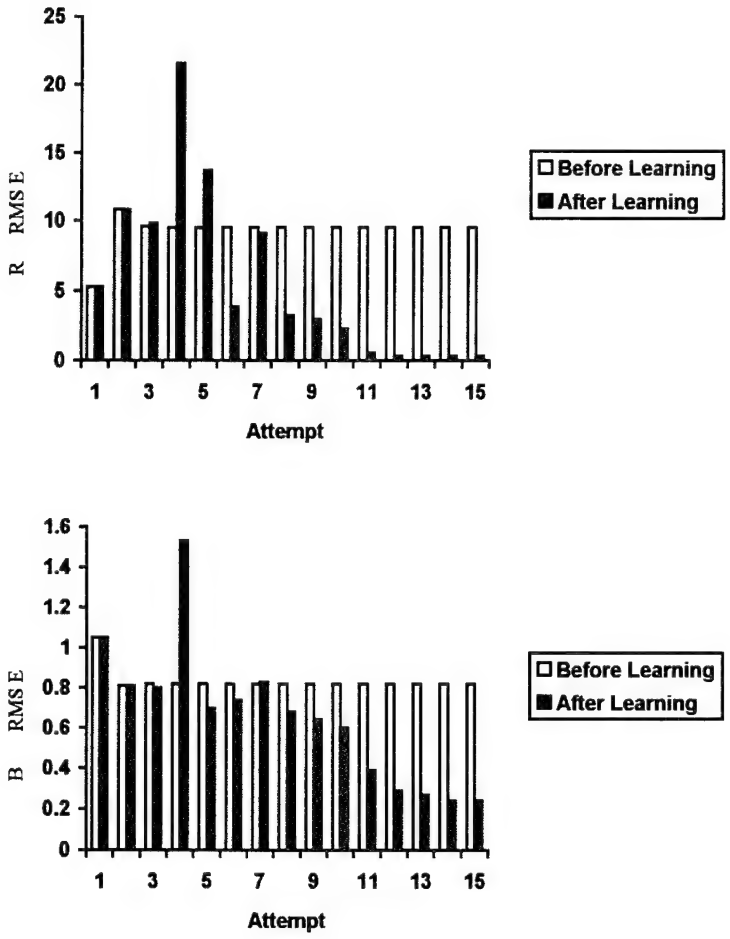


Figure 3.8. *Improvement in chasing behavior due to learning.*

Learning from rules how to fly in formation

Experiment 3b uses a neural network to improve tree avoidance during formation flying. As shown in Fig. 11, the network has eight inputs representing range, bearing, and rates associated with the jet's mate and the closest tree. One net output learns to turn the jet (yawThrust), while the other output learns to thrust the jet laterally (latThrust). Initially, these nets contribute nothing to the overall control command (all their weights are zero). Over time, however, they discover how to steer the jet past trees while keeping it in formation.

The goal of neural net learning is to reduce the control commands suggested by the tree-bypassing rule (which only considers range, bearing, and rates associated with the closest tree). Because the net has more context information than any single rule, it can discover control interactions that augment the ability of the rules.

In this experiment, the neural net discovers an interesting solution. As shown in Fig. 3.10, it transforms the original triangular flying formation into an "L" formation, where one wingman is directly behind the chaser, and the other is off to the side.

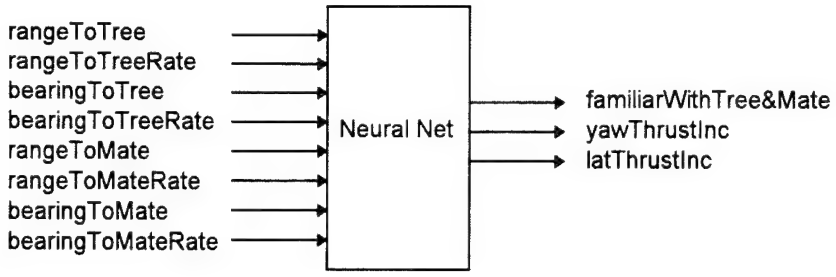
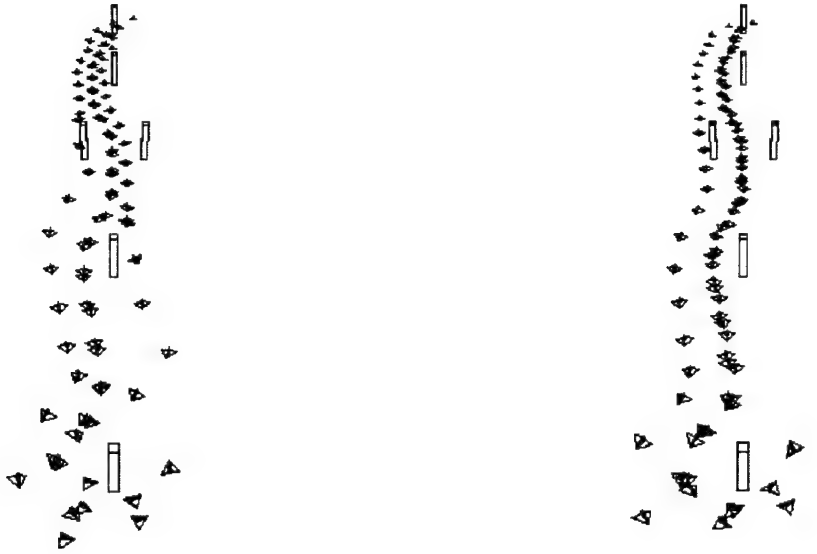


Figure 3.9. *Neural network used to improve formation flying behavior.*

Figures 3.11 and 3.12 show that the learned formation allows the jets to stick together while flying through the forest. In essence, the neural net used the rule-based control law as a good starting point, but improved upon it in a non-obvious, yet efficient, way.

(a) *Before learning.* (b) *After learning.*

Figure 3.10. *Comparison of flying formation before and after learning.*



(a) *Before learning, 6th attempt.* (b) *After learning, 6th attempt*

Figure 3.11. *Snapshots of flying formation behavior before and after learning.*

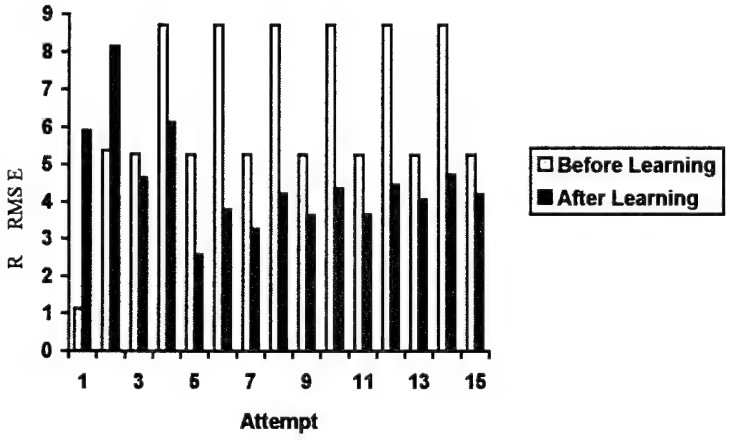
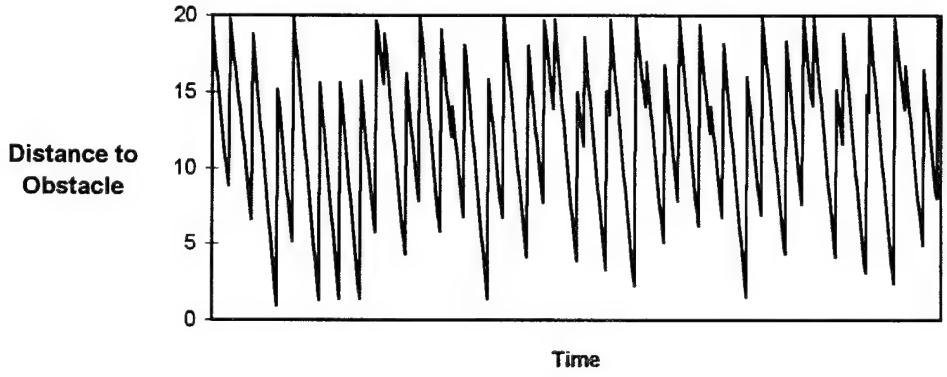


Figure 3.12. *Improvement in formation flying due to learning.*

Learning from operator how to avoid obstacles

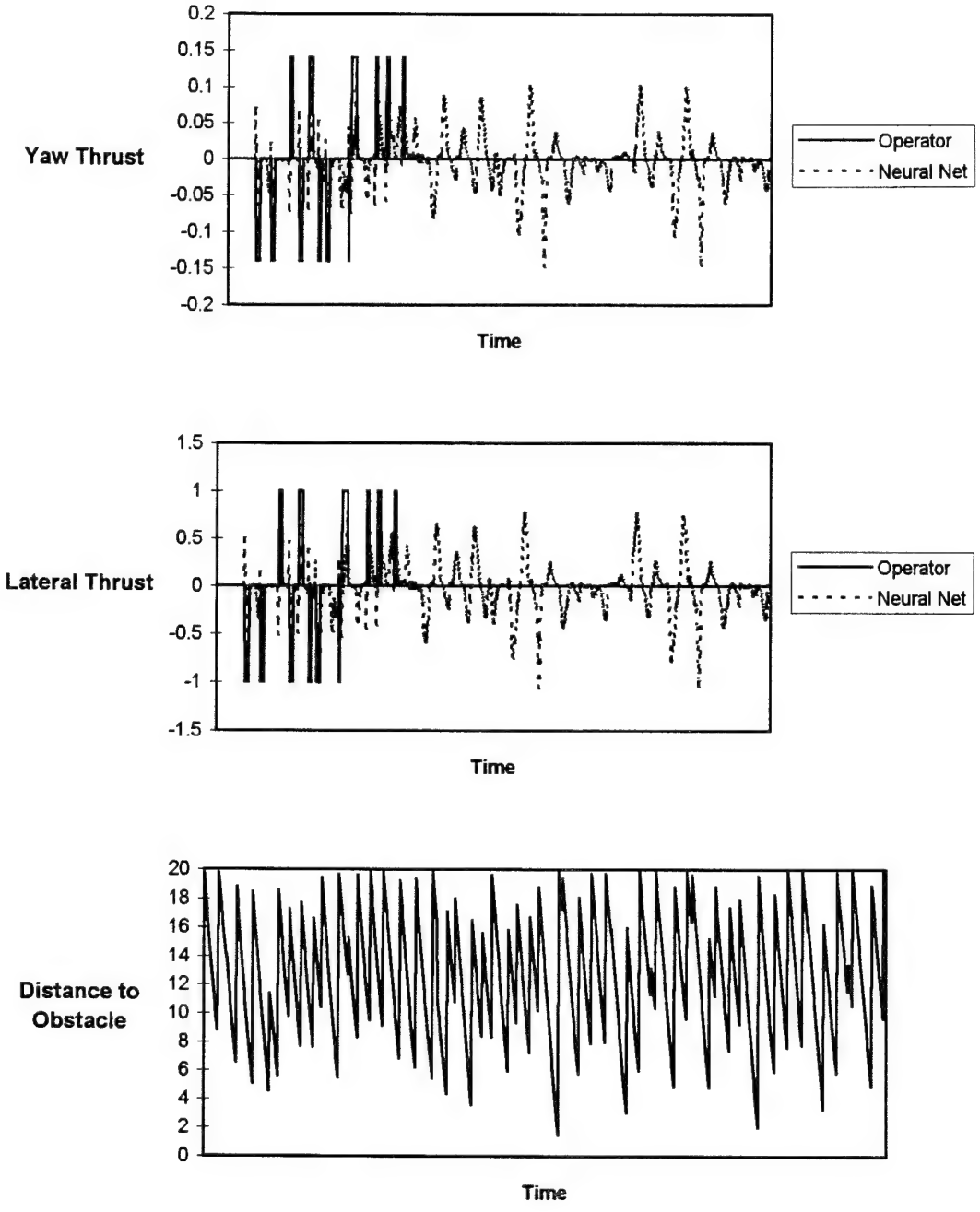
In Experiment 4, on-line teleoperator inputs were used to teach the retreating tank how to avoid trees. The retreating tank starts out using rules to go from waypoint to waypoint, but no knowledge of how to avoid trees. A neural net adds yaw thrust and lateral thrust to avoid trees as it learns.

The net starts out contributing nothing. Figure 3.13a shows the retreating tank's distance to obstacles (trees) without learning. The plot only includes data for obstacles that are within 45 degrees of the tank's velocity vector and less than 20 meters away. Figure 3.13b shows how learned teleoperator control commands increase the minimum distance to obstacles.



(a) *Before learning.*

Figure 3.13. *Continued on next page.*



(b) *After learning.*

Figure 3.13. *Improvement in retreating tank distance to obstacle due to on-line operator training. Operator teaches retreating tank how to avoid obstacles using yaw thrust and lateral thrust. Closest distance to tree decreases over time due to learning as neural net takes over responsibility from operator.*

3.3.2 Bridge Crossing Battle Scenario

The second battle scenario involves getting a CGF to cross a bridge efficiently. It is assumed that the bridge has a north end and a south end, and that the tank can focus on waypoints located at both ends of the bridge and can focus on its centerline (a "path"). Figure 3.14 shows the simplified environment of the bridge crossing scenario.

Note that in the forest chase experiments, the goal of learning was to improve the performance of an avoidance behavior, that is, the avoidance of obstacles. This experiment involves behavioral attraction, particularly the regulation of movement with respect to the centerline of a bridge. The main goal is to is to have the tank cross the bridge along its centerline when approaching it from either the right or the left. It must start out using rules only, and should then improve its performance given additional corrective rules and on-line teleoperator inputs.

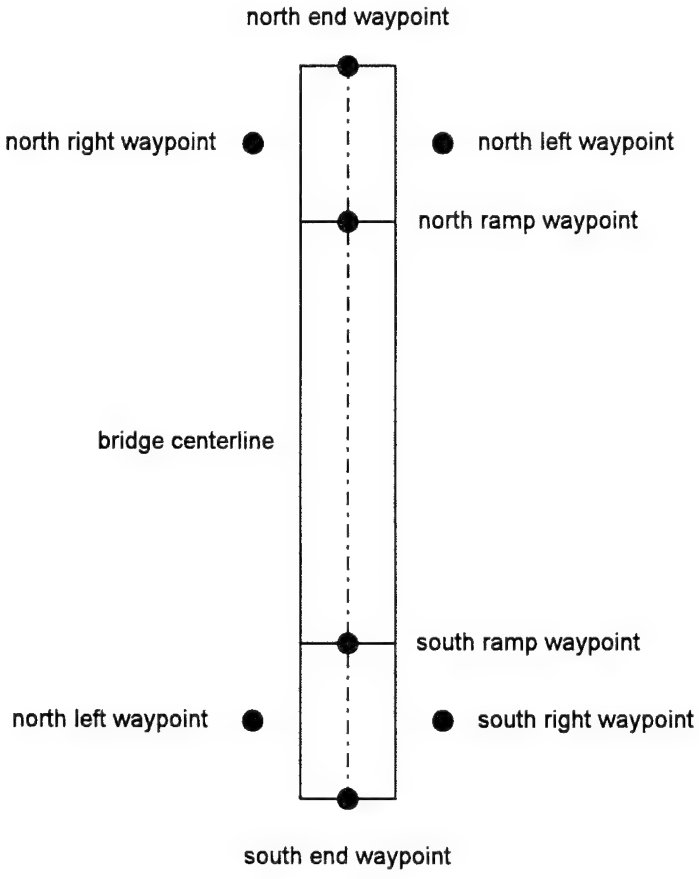


Figure 3.14. *The simplified bridge crossing environment.*

CGF Performance with No Learning

There are 27 rules that make up the bridge-crossing knowledge base. As described for the forest chase scenario, they implement an intuitive, modular control law. Shown below is an outline of the rule heirarchy.

Solo Behaviors

LongitudinalSpeed()
    LonThrust()
LateralSpeed()
    LatThrust()
YawSpeed()
    YawThrust()
GetOperatorCommands()

Environmental Focus

WaypointFocus()
    rangeTo
    rangeToRate
    bearingTo
    bearingToRate
PathFocus()
    rangeTo
    rangeToRate
    bearingTo
    bearingToRate
    bearingFrom
    bearingFromRate

Environmental Behaviors

CrossBridge()
    ApproachBridge()
        FollowWaypoints()
            GoToWaypoint()
            LookAtWaypoint()
            LateralSpeed()
        FollowPath()
            FollowPathWithRules()
                GetToCenterLine()
                    GoToPath()
                    AlignWithPath()
                StayOnCenterLine()
                    GoToPath()
                    AlignWithPath()
            FollowPathWithOperator()
                GetOperatorCommands()
            FollowPathWithNets()

```
                                    LearnToFollowPath()
                                        NetLocate(0
                                        NetShape()
                                    RecallHowToFollowPath()
                                        NetLocate()
                                        NetRecall()
GoOverBridge()
        FollowWaypoints()
        FollowPath()
DepartBridge()
        FollowWaypoints()
TurnAround()
        FollowWaypoints()
```

Bridge crossing, no learning

The nominal function of the bridge-crossing rules is to follow waypoints that are located at each end of the bridge. Waypoints are followed by focusing on the next waypoint, looking at it, and moving forward at a certain speed. Because there is no notion of alignment with this preliminary strategy, if the tank approaches from the left or the right, and not straight on, it will produce an arcing trajectory in it's attempt to go from it's beginning waypoint to the one on the other side of the bridge. Figure 3.15 shows the resulting overshoot of the tank's distance from the bridge centerline.

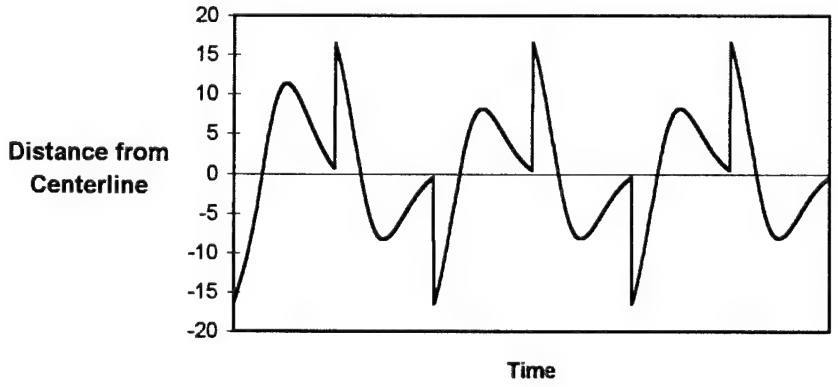


Figure 3.15. *Nominal distance of tank from centerline of bridge using waypoint-following rules only. Plot shows six approaches, with approach direction alternating from right to left.*

CGF Performance with Learning

A learning neural net can learn to reduce this centerline overshoot. As shown in Fig. 3.16, the neural net takes in range and bearing from the path (centerline), and outputs yaw and lateral thrust commands. The net is trained by rules and teleoperator inputs.

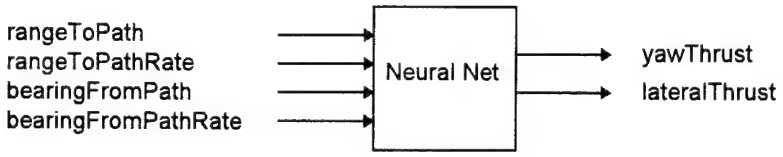


Figure. 3.16. *Neural net used by rules and operator to improve bridge-crossing performance.*

Learning from rules how to improve bridge crossing

Corrective rules use the GoToPath() and AlignWithPath() library functions to provide yaw and lateral thrust commands that are used to help control the tank, and train the neural net. Figure 3.17 shows the improvement in performance based on rule-based net training.

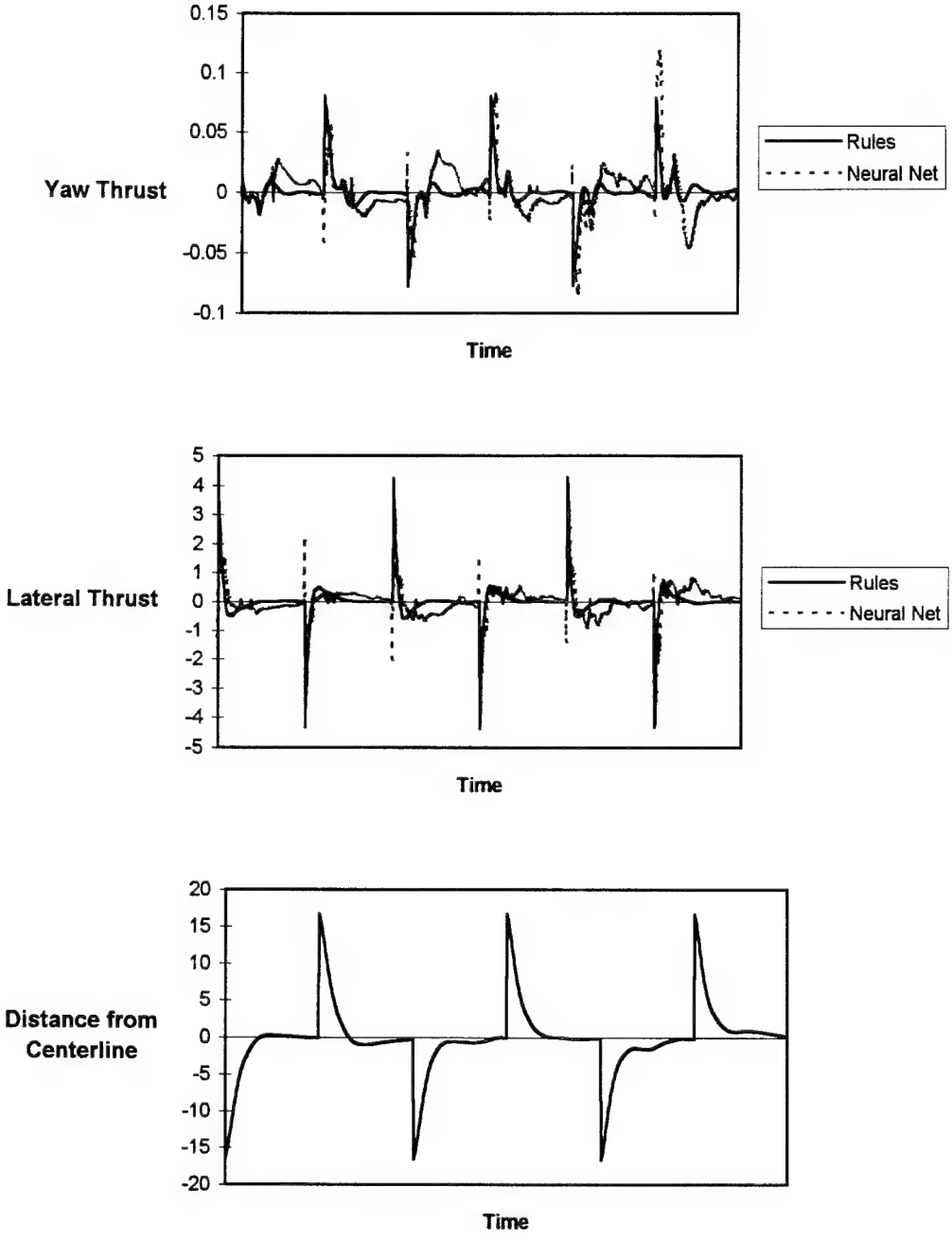


Figure 3.17. *Improvement in bridge-crossing performance due to neural net learning from path-following rules. Rules use vehicle library functions GoToPath() and AlignWithPath() to provide longitudinal thrust, lateral thrust, and yaw thrust corrections that are used for learning.*

Learning from operator how to improve bridge crossing

In this experiment, corrective yaw and lateral thrust commands provided by an operator are used to help control the tank, and train the neural net. Figure 3.18 shows the improvement in performance based on rule-based net training. By controlling yaw thrust and lateral thrust independently, the operator can vary the learned response. For example, the data shown here indicates that the operator taught the tank to "broadslide" into the turn by laterally thrusting toward the bridge centerline while turning away from it. The last two of the six bridge crossings shown in Fig. 3.18 are done without any operator input, showing that the neural network has learned to improve significantly the initial performance shown in Fig. 3.15.

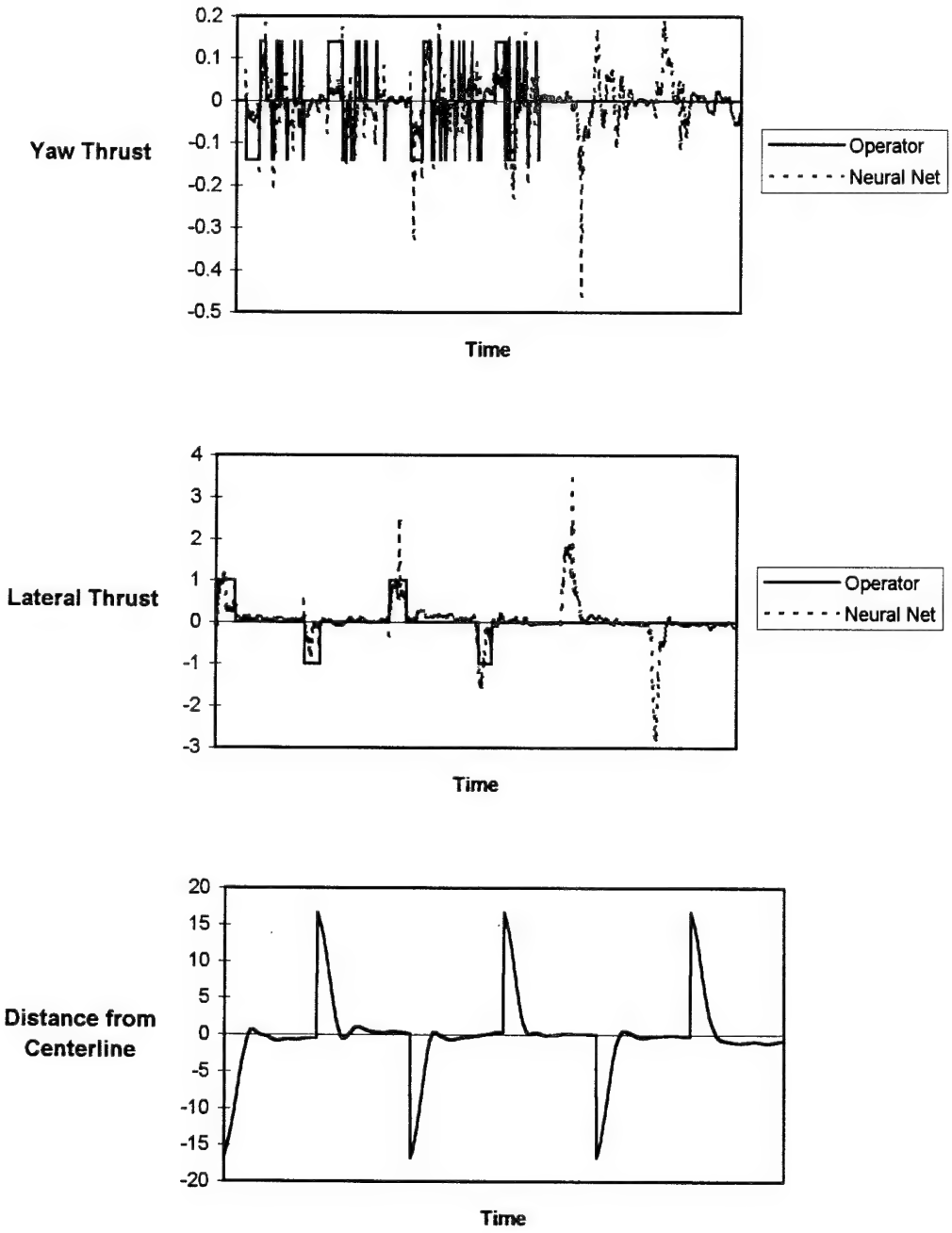


Figure 3.18. *Improvement in bridge-crossing performance due to neural net learning from on-line operator. Operator controls independent yaw thrust and lateral thrust.*

These results indicate that CGFs can not only initiate tasks using a predominately rule-based task description, but can improve their performance using corrective rules and teleoperator inputs on-line. Furthermore, because the behaviors are developed with respect to a focus of attention, the acquired skill can be re-applied to other situations that involve similar objects simply by resetting the CGF's focus of attention.

3.4 INTEGRATION OF INTELLIGENT CGF BEHAVIORS WITH EXISTING SAF SYSTEMS

3.4.1 Related Work

The secret to this type of machine learning is neural networks. Neural networks are function approximators that learn to represent meaningful information by being shown typical input/output examples. Different applications require different neural network architectures.

Semi-Automated Forces

SIMNET SAF: The SIMNET SAF system was the first real-time SAF system developed for combined arms man-in-the-loop training. First developed in 1985, the architecture consisted of an "AI" frontend running on a Symbolics Lisp machines, with the simulation backend running on Masscomps. This architecture was deficient in that the interface between the Symbolics and Masscomps was not flexible. Once a mission was downloaded from the frontend to the backend, the mission could not be modified. This architecture was later migrated to a MIPS frontend and backend. If this architecture is chosen as the basis for Phase II, the NeurRule Technology will interface with the models on the Simulation Computer (Backend) and enhance their tactical capability. Decisions which were static and algorithmic up to now, will be made adaptive and reactive. Several of the Phase I team members developed portions of this system.

ModSAF: ModSAF is the fundamental SAF layer advocated by the Government. It arose out of the original SAF architecture, and is DIS compatible. The purpose of the architecture is to provide modularity between different functional components of the SAF system. It has not, however, improved on the performance of the planning modules themselves. If ModSAF is chosen as the basis for Phase II, the NeurRule Technology will be used to replace those portions of the simulation software which control entity behavior. This technology will save the cost of maintaining and adding functionality to ModSAF at the entity model level.

WISSARD-SOAR: The SOAR model of reasoning uses a rule-based technology similar to that used by KATrix. However, it lacks the neural network learning capability. KATrix's neural network integration methodology could make SOAR run faster for commonly used air-to-air maneuvers.

WISSARD-FFCS: The Fast Futures Contingency System (FFCS) uses an event tree to generate potential outcomes. KATrix's technology can generate candidate trees which represent better approximations of outcomes.

Previous Applications of the NeurRule™ Technology

We have applied the concepts embodied in the NeurRule Control Architecture to a variety of control tasks.

- Handelman and Lane [1993a] address the simple problem of learning how to balance and position an inverted pendulum. Control system processing transitions smoothly and quickly from declarative rules to reflexive neural networks while significantly improving performance and decreasing computation.

- In Handelman [1992] teleoperator inputs are used to download a person's hand/eye coordination into a learning control system. Initially, because the neural network starts out knowing nothing, the operator must perform the pole-balancing task. Over time the neural network assumes more and more control responsibility, and ends up encoding an effective nonlinear control law that is very different from the operator's piecewise linear inputs.

- In Handelman, Lane, and Gelfand [1992] rules that are intentionally similar to what a flight instructor might tell a student pilot are used to train neural networks how to perform a straight-in aircraft approach and landing. This allowed a smooth transition from rule-based to network-based operation results in improved performance and reduced computation.

- Handelman, Lane, and Gelfand [1993] and Handelman and Lane [1993b] describe a robotic assembly task in which the controller uses intuitive rules to begin the task, and neural network learning to improve system performance.

- In Handelman and Lane [1993c], the smart opponent concept is applied to an interactive video game. The computer generated smart opponent learns a player's context-dependent moves and uses them against him.

Other Approaches to Hybrid Intelligent Agents & Learning

The uniqueness of our approach to robotic skill acquisition can be seen when compared to other efforts in computer generated forces and behavioral representation. Many researchers point out the need for both high-level symbolic processing and low-level reactive behavior in intelligent agents [Gordon & Subramanian, 1993; Gat et al, 1993; Chaib-draa et al, 1993]. However, few systems smoothly integrate the two types of processing, and even fewer systems learn and adapt.

Our approach to reinforcement learning also is consistent with both classical and genetic algorithmic [Stengel, 1986; Fogel, 1995 and Goldberg, 1989] approaches in that it can determine optimal control laws and plans of action. However, both classical optimization and genetic algorithm approaches often slowly converge to such optimal solutions. This is usually due to poor choices for the initial parameter values and relatively undirected (i.e. non-gradient) searches in fairly large parameter spaces. Since the NeurRule Technology presented here benefits from the use of both human explanation and demonstration, convergence to the optimal solution during the reinforcement learning process is greatly accelerated.

3.4.2 Role of NeurRule™ Technology

Much effort has been devoted to *planning* in SAF systems, especially with respect to vehicle movement. ModSAF's architectural paradigm in planning movement is that of generate and test. Several possible paths are generated and are tested for acceptability based on a few simple factors. A second pass categorizes the best approach. When standards are not reached, the better approach is edited to improve its acceptability. In this way, a vehicle can be made to avoid a building, cross a bridge, etc.

Our approach to CGF behavior includes the ability to easily implement *pre-determined* plans of action with nearly verbal rule-based task descriptions. The difference is between teaching a pilot how to land a helicopter by giving him a flight manual and interactive help (something the NeurRule technology is good at), and in telling a pilot to plan a flight from home base to a target while avoiding known surface-to-air missile sites (something existing ModSAF technology is good at). Both technologies can coexist, and in fact can benefit from each other.

NeurRule rule-based control technology keeps the control law in a form that is easily modified by system operators. Algorithms and equations of the ModSAF planning modules can be accessed through the rules, and need not be edited directly. Such interaction is an intentional architectural feature of ModSAF. For example, the LibMoveMap navigational algorithms in ModSAF include functions permitting the path search process to be modified and interrupted during operation (as evidenced by the functions movemap_set_goal(), movemap_update(), movemap_next_point(), movemap_change_obstacle(), etc.). In fact, the *LibMoveMap Programmer's Guide* states,

> "LibMoveMap has been designed to solve the near-term navigation problem ... In order to make the problem more tractable, the following problems are not addressed by libMoveMap, and thus must be addressed by higher layer software modules: Selection of non-conflicting near term goals, choosing new goals when the desired behavior cannot be achieved, implementing the plans produced by giving command to the vehicle dynamics model, ..."

Application of the NeurRule Technology proposed in this project can fill this gap. Futhermore, the type of on-line neural network learning proposed provides additional learning capabilities not yet exploited in any SAF system to date, including:

- Behavior blending. Neural nets can be used to "fuzzify" rules of engagement which decide how to switch from one task to another. For example, assume that there exist SAF "attack" and "retreat" behaviors. Some means must be provided for switching between the two behaviors. Neural nets can learn when to invoke each behavior based on experience.

- Context dependent behavior. What the CGF does depends on what other entities, real and synthetic, do on the virtual battlefield.

- Different training experience for different soldiers. Different soldiers are apt to do different things, and the context-dependent behavior will reflect these differences.

- Different training experience for a given soldier who does different things over time.

- Reduced need for discrete levels of competency in opposing forces resulting in more realistic simulations. There need not be abrupt jumps in the perceived ability of a synthetic opposing force. No level 1 difficulty, then level 2 difficulty, etc., just a smooth and steady (more human-like) increase in CGF ability.

- Battle trainers can create personal CGFs that evoke specific behaviors (*very* aggressive, *very* incompetent, etc.) favored by them.

With the simple ModSAF hooks described above, these capabilities can augment existing CGF behaviors.

NeurRule Technology also can help alleviate some known problems with existing SAF systems. As an example, the *SAFDI User's Guide* points out that, "as with any complex software, the SAFDI system has limitations, both in terms of behavioral representation and realism." We don't point out some of these faults to beat up on SAFDI, but to indicate how NeurRule technology might be used to fix these and similar SAF problems.

Here are some hints given by the *SAFDI User's Guide* about ***dynamic obstacles*** [page 4-4]:

> "SAFDI entities never consider vehicles which are in motion at the moment they route. As a result, it is up to the operator to avoid collisions with moving vehicles. Keep SAFDI units well away from each other and manned simulators when in motion. When moving with manned simulators, let the manned simulators move behind the SAFDI entities so they can see them and help to avoid collisions. Finally, the SAFDI Operator must carefully watch entities that are routing near other vehicles at all times and either halt or reroute entities that threaten to collide with one another."

NeurRule CGFs are capable of focusing on many things at once, such as the closest enemy, and a particular mate, and the closest tree. Using low-level inhereted behaviors, agents can easily be made to track or avoid any of these things. In fact, using the learning ability of neural nets, agents can learn the dynamics of other entities in the environment and predict their paths. NeurRule

technology can therefore be used to modify and improve upon existing SAF obstacle avoidance behaviors, and in doing so reduce operator workload.

Here are some hints given by the *SAFDI User's Guide* concerning ***formations*** [page 4-4]:

> "Normally, SAFDI entities are not aware of, and therefore will not maintain, formations. The only exception to this is when SAFDI units are ordered to attach and follow, in which case they appear to maintain a formation relative to the leader. If the leader moves erratically, the formation will break up. Since SAFDI entities do not avoid other moving vehicles, the SAFDI Operator should take care to use attach and follow only when the leader and all of the following vehicles have sufficient distance between them. As always, the SAFDI Operator should observe the motion of the following entities and alter their routes if a collision is imminent."

As demonstrated by the Phase I results described in Section 1.6, NeurRule agents possess data and behaviors capable of not only maintaining basic formations with no operator intervention, but can even learn on their own how to improve on an initial rule-based task description. By using this technology, formations that would otherwise break down into anomalous behavior would be fixed, and operator workload would be reduced.

The NeurRule Intelligent Agent software libraries are written in C. They are relatively efficient, and completely portable. Libraries are running on PC-compatible computers under both DOS and Windows, and on Silicon Graphics Workstations under IRIX 5.2. Ports to other operating systems and machines are straightforward. Consequently, the technical feasibility of linking the NeurRule Intelligent Agent software libraries with SAF systems written in C or C++ is assured.

4. PHASE II WORK

4.1 PHASE II TECHNICAL OBJECTIVES

The two main goals of the Phase II effort are to demonstrate that:

- KATrix's NeurRule Intelligent Agent Technology enables fully-autonomous and adaptive CGFs to be developed easily, quickly, and efficiently by non-programmers for dismounted infantry applications in urban environments, and
- SAF operators can gradually shift their attention from low-level control tasks to high-level coordinated plans of action as smart opponent and virtual teammate CGFs gain competence learning relevant assault tactics and evasive maneuvers.

Specific Phase II objectives include:

- Develop a set of "core" NeurRule Technology CGF behaviors (e.g. attack, hide, evade, formation, etc.) consistent with operations of dismounted infantry in an urban environment.
- Create a military-friendly NeurRule Technology CGF Development System, with a point-and-click graphical user interface, allowing SAF operators to quickly design complex CGF behaviors by selecting and connecting appropriate core (and custom) behaviors and instantiating them with desired goals and constraints of operation.
- Integrate the NeurRule Intelligent Agent Control Architecture with the Dismounted Infantry in a Virtual Environment (DIVE) simulation system. Integrate with an existing dismounted infantry SAF (e.g. SAFDI) if appropriate.
- Interface the NeurRule Technology CGF Development System with the DIVE simulation system to enable interactive CGF behavior creation and modification.
- Prototype Smart Opponent and Virtual Teammate CGF behaviors
  - Generate intelligent CGF behaviors implemented in Phase I using Phase II Development System.
  - Demonstrate ability of SAF operator to both specify smart CGF behaviors and fine-tune using cooperative and reinforcement learning.
  - Assemble CGF behaviors developed into tactical libraries for dismounted infantry operations in urban environments.
- Quantify power of NeurRule approach to CGF Development in terms of:
  - Reduced cost of developing CGF behavior

- Reduced operator workload
- Reduced number of operators required to obtain desired level of simulation fidelity.

4.2 PHASE II TECHNICAL APPROACH

4.2.1 Authoring of Intelligent Computer Generated Force Behaviors

KATrix has currently developed a NeurRule™ Technology Authoring Tool that permits a designer or operator to graphically flowchart desired behaviors, specifically, the interaction of intelligent agent reflexes, goals, and skill learning with player responses and other game control logic. The graphical tool simplifies the overall development process by permitting designers to schematically build complex intelligent agent behaviors incrementally, without a lot of programming. The NeurRule Authoring Tool provides a framework within which NeurRule Intelligent Agents may be created. The Tool promotes a three-step design procedure, depicted in Fig. 6.1. First, agent classes are created. Then behaviors for these agents are created. Finally, agent behaviors are tested and tuned within the host application.

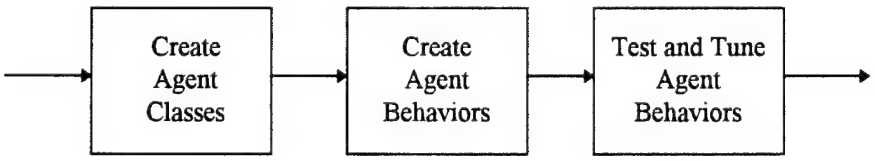


Figure 4.1. *Three step design procedure for building NeurRule Intelligent Agents.*

Agent classes represent the various types of entities that can interact within the host application. Users create new agent classes by deriving them from a set of base agent classes.

Agent classes have data and behaviors associated with them, and derived agent classes inheret data and behaviors associated with the agent classes from which they are derived. By mirroring the polymorphism and inheritance features of object-oriented programming technology, agent classes represent a compact and flexible entity description.

Using the NeuRule Authoring Tool, users interact in point-and-click fashion with the agent class hierarchy. Depicted in Fig. 4.2, the agent class hierarchy shows the ancestry of all existing classes. For example, at the root is the ENTITY agent class, from which all other agents classes are derived. An ENTITY has a position and orientation (among other things), and behaviors such as SetOrientation. A MOVING_ENTITY is derived from an ENTITY, and therefore has all of its capabilities, but includes additional information and capabilities relevant to entities that move,

such as a velocity and behaviors like GoTo and LookAt. Similarly, agent classes such as VEHICLE and PERSON are derived from MOVING_ENTITY.

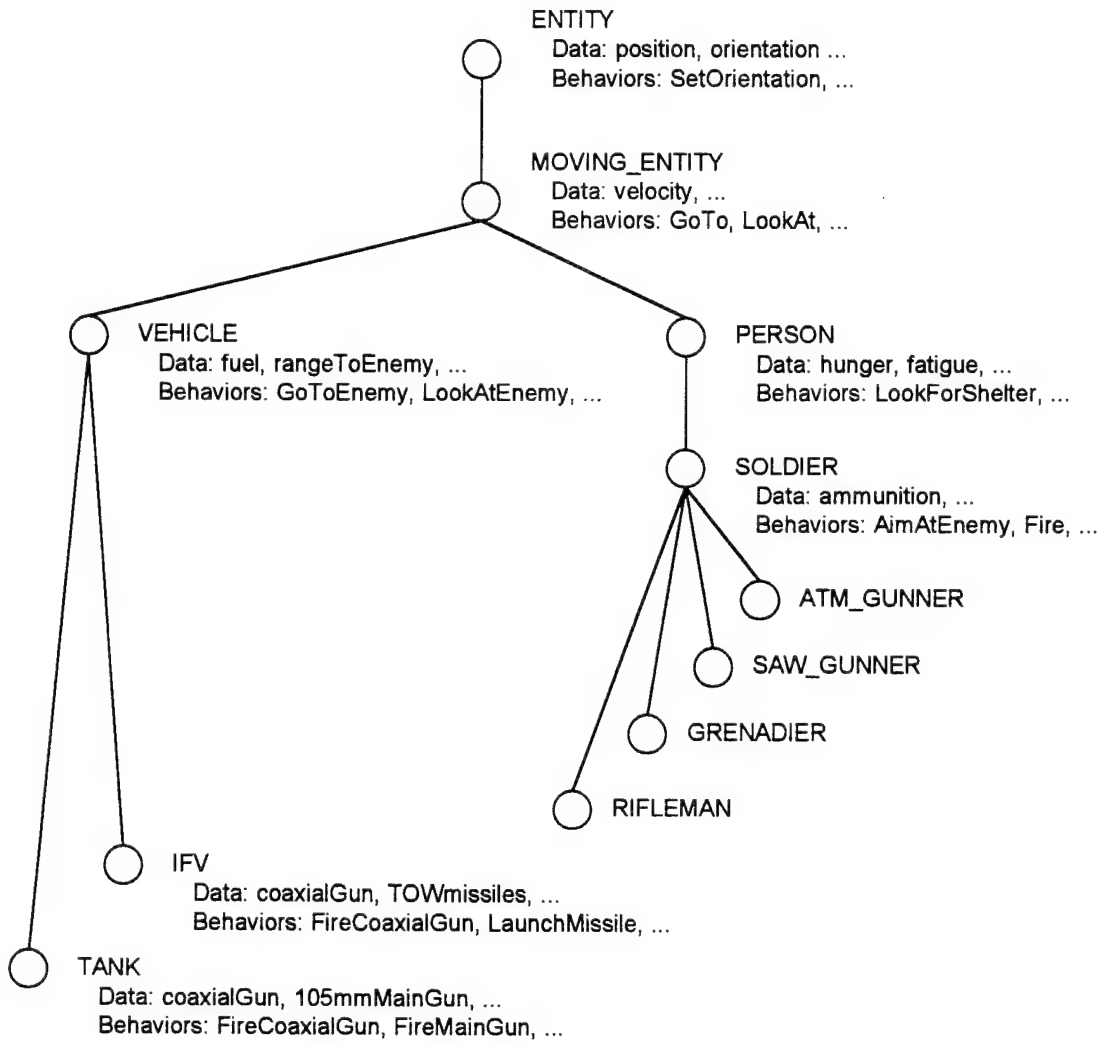


Figure 4.2. *Sample agent class hierarchy.*

Although the NeurRule Authoring Tool includes many agent classes implementing common entities, the intent is to make it easy for users to add new agent classes that reflect the specific needs of their application. Fig. 4.2 depicts how TANK and IFV agent classes might be derived from the VEHICLE agent class, whereas the RIFLEMAN, GRENADIER, SAW_GUNNER, and ATM_GUNNER agent classes might be derived from a SOLDIER class which is derived from the PERSON class.

The availability of numerous inherited behaviors greatly simplifies the creation of new intelligent agent behaviors. For example, in the simulations discussed previously (Section 3.3),

the JET agent class was derived from the VEHICLE agent class, a base class of the library. Consequently, the JET behavior FollowMate invokes the basic VEHICLE behaviors GoToMate, AlignWithMate, LookInDirection, LongitudinalSpeed, and LateralSpeed.

In order to create an agent behavior, the user must give the behavior a name, specifiy the agent's focus of attention for that behavior, and build up the desired behavioral response using rules, neural nets, predefined low-level core behaviors inherited from ancestral agents, and predefined high-level behaviors called "ClipSmarts".

4.2.2 Trainer-in-the-loop CGF Behavior Development

To manage the increased complexity associated with urban environments, the user interface to a SAF system must not only allow the operator individual control over all entities in the simulation, but also enable him or her to interactively assemble libraries of compound or coordinated actions, specify the goals of the behavior, and automatically trigger appropriate patterns of response given the context of the situation. In this way, the SAF operator can gradually devote his or her attention to higher-level control tasks and mission responsibilites as the CGF intelligent agent gains competence implementing lower-level control tasks, coordinated actions and context-based behavioral responses.

The key to obtaining realistic synthetic soldiers is to combine the booksmarts contained in Handbooks and Manuals with the operational experience of real soldiers. Insofar as battle trainers are the fighting experts, the goal is to transfer their knowledge of fighting into synthetic soldiers (and hence, of course, into real soldiers as well). Figure 4.4 represents how a battle trainer / SAF operator fits into the CGF behavior development cycle.

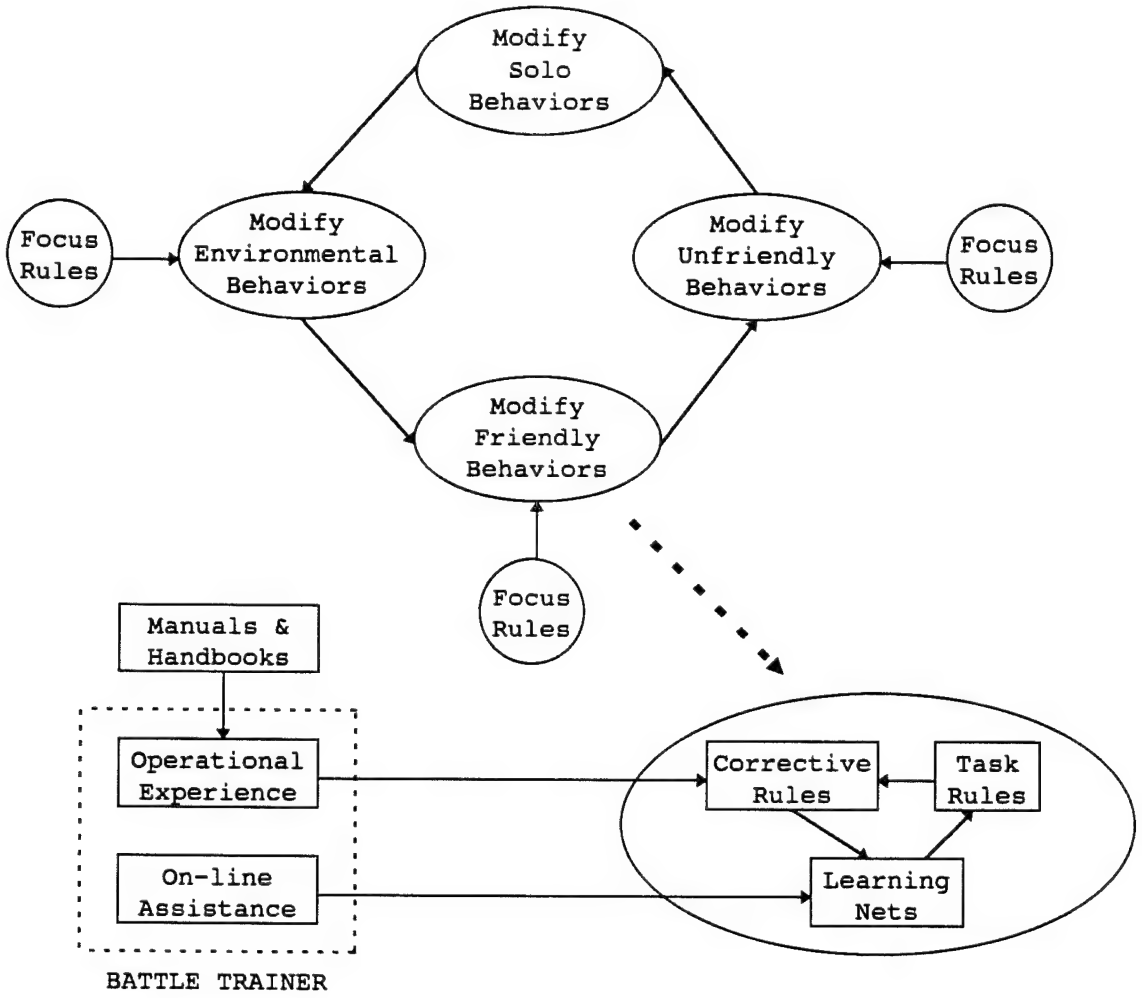


Figure 4.4: *Integration of battle trainer into CGF behavior development cycle.*

Much of the Phase II work will involve creating convenient channels of communication whereby an operator can *quickly* transfer rule-based task knowledge and satisfaction-based corrections to a synthetic soldier.

5. CONCLUSIONS

5.1 IDENTIFIED NEED MET BY THE PROJECT

The advantages of computer generated forces capable of acquiring skill on-line are well documented. For example, Deutsch [1993] points out that, "Second, third, and fourth encounters must present responses which keep the trainees focused on the domain problem, and not on victory by exploiting the computational solecism of the computer generated adversary ... Adroit but predictable responses are doomed." Downes-Martin [1993] states that, "Given the existence of current CGF systems ... it is reasonable to propose a system in which each human decision maker is responsible for the long term training and use of his or her own software assistant. These software models are initially trained in faster than real time simulations against each other and more advanced software, with human supervision." We believe our approach to developing intelligent computer generated forces enables these capabilities.

5.2 ANTICIPATED RESULTS

The overall goal of this project is to demonstrate that KATrix's NeurRule Technology can augment current SAF capabilities by integrating rule-based systems with neural networks, allowing "smart opponents" and "virtual teammates" to be created that continuously learn from interactions with manned forces, human instructors, and their own mistakes. The Phase I results demonstrate the effectiveness of the NeurRule Technology. Phase II will make this technology easy to use by SAF operators and military personnel through an intuitive point and click graphical user interface that enables the SAF operator to quickly design and assemble complex intelligent behaviors from the individual soldier up to the platoon level by selecting and connecting appropriate "core" (follow, avoid, group, formation, attack, defend, cautious, aggressive, etc.) or customed-designed behaviors and instantiating them with the desired goals and constraints of the operation. Successful completion of the project will result in semi-automated forces for distributed interactive simulations that:

- ❑ Can continuously adapt their behavioral models in real-time.
  - From its own behavior.
  - From its opponents (both successful strategies and mistakes).
- ❑ Can be shown and told by a human instructor.
  - Cognitive plans of action.
  - Hand/eye coordination.
- ❑ Can be created by the SAF operator using a GUI Development Tool instead of by a programmer.
- ❑ Have user selectable learning rates and level of competencies.
- ❑ Reduce their computational burden as the CGF entity learns.

5.3 POTENTIAL USE BY THE FEDERAL GOVERNMENT

Integration of the KATrix's NeurRule Technology with existing simulation and training systems will make CGF behaviors more realistic and adaptable. In addition, this also will save the government money in costly repetitive software modifications by enabling SAF operators to modify SAF behavior without the intervention of programmers. The NeurRule Technology is applicable to Army SAF systems (CCTT), aviation SAF systems (AVCATT, WISSARD, TACTS range, etc.), undersea training, weapons development (War Breaker), etc.

There is a great effort underway to provide pilots, tank commanders, etc., with pilot's associates. These associates are computer programs which help the pilot quickly assess his situation and make decisions. A hybrid neural-network/rule-based pilot's associate would get better at recognizing situations over time. These SAF-based intelligent entities could be trained in the laboratory and then installed in a live combat vehicle as a pilot's associate. Eventually, the entities can become smart enough to pilot the vehicles alone. They would be able to pilot unmanned ground vehicles, air vehicles, or underwater vehicles. Just like soldiers, the entities can learn from their mistakes while performing missions. These learned lessons can be assimilated into the knowledge bases of other entities.

6. REFERENCES

Albus, J., 1975. "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *J. Dyn. Syst. Meas. Control*, Vol. 97, pp. 270-277.

Chaib-draa, B., Paquet, E., and Lamontagne, L., 1993. "Integrating Reaction, Planning and Deliberation in Architecture for Multiagent Environment," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Orlando, Fla., March, pp. 45-55.

Deutsch, S., 1993. "Notes Taken on the Quest for Modeling Skilled Human Behavior," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Orlando, Fla., March, pp. 359-365.

Downes-Martin, S.G., 1993. "Collaborative Distributed Simulation," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Orlando, Fla., March, pp. 131-142.

Fogel, D., 1995. *Evolutionary Computation - Toward a New Philosophy of Machine Intelligence*, IEEE Press, New York.

Gat, E., Fearey, J., and Provenzano, J., 1993. "Semi-Automated Forces for Corps Battle Simulation," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Orlando, Fla., March, pp. 69-74.

Goldberg, D., 1989. *Genetic Algorithms - In Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., New York.

Gordon, D.F., and Subramanian, D., 1993. "A Multistrategy Learning Scheme for Assimilating Advice in Embedded Agents," in Michalski, R.S., and Tecuci, G., Eds., *Proc. 2nd Int'l. Workshop on Multistrategy Learning*, George Mason University, May, pp. 218-233.

Handelman, D.A., 1992. "Pinhead," a computer exhibit in "It's All in Your Head," a 3-year, 7-city traveling exhibition on the brain developed for the Franklin Institute Science Museum, Philadelphia.

Handelman, D.A., and Lane, S.H., 1993. "Fast Sensorimotor Skill Acquisition Based on Rule-Based Training of Neural Networks," in Bekey, G., and Goldberg, K., Eds., *Neural Networks in Robotics*, Kluwer Academic Pub., Norwell, Mass., pp. 255-270.

Handelman, D.A., and Lane, S.H., 1995. "Human-to-Machine Skill Transfer Through Cooperative Learning," in Gupta, M.M., and Sinha, N.K., Eds., *Intelligent Control Systems - Theory and Applications*, IEEE Press, Piscataway, New Jersey, pp. 187-205.

Handelman, D.A., and Lane, S.H., 1993. "Smart Virtual Opponents with Human-Like Learning for Interactive Computer Entertainment," presented at the *1993 Meckler Conference on Virtual Reality*, New York, August.

Handelman, D.A., Lane, S.H., and Gelfand, J.J., 1990. "Integrating Neural Networks and Knowledge-Based Systems for Intelligent Robotic Control," *IEEE Control Systems Magazine*, Vol. 10, No. 3, April, pp. 77-87.

Handelman, D.A., Lane, S.H., and Gelfand, J.J., 1992. "Robotic Skill Acquisition based on Biological Principles," in Kandel, A., and Langholz, G., Eds., *Hybrid Architectures for Intelligent Systems*, CRC Press, Boca Raton, Florida, pp. 301-328.

Handelman, D.A., Lane, S.H., and Gelfand, J.J., 1993. "Prospects for Cooperative Learning in Intelligent Vehicles," *Proceedings of the 1993 IEEE Regional Conference on Control Systems*, New Jersey Institute of Technology, August.

Lane, S.H., Handelman, D.A., and Gelfand, J.J., 1992. "Theory and Development of Higher-Order CMAC Neural Networks," *IEEE Control Systems Magazine*, Vol. 12, No. 2, April, pp. 23-30.

Miyamoto, H., Kawato, M., Setoyama, T., and Suzuki, R., 1988. "Feedback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator," *Neural Networks*, Vol. 1, pp. 251-265.

Stengel, R., 1986. *Stochastic Optimal Control - Theory and Application*, John Wiley and Sons, New York.